Optimizing Recognition Representation for Use in Anomaly Detection

Jialin Yu[1]

MSc Computer Graphics, Vision and Imaging

Supervisor: Lewis D. Griffin

Submission date: 05 September 2019

Abstract

Anomaly detection (AD) is a task of finding instances of nonconformity from an analyzed data set. Transfer learning, which refers to applying model trained from a richer dataset into a database with insufficient examples for cross-validation, has been suggested to contribute to the issue of model underfitting in a smaller dataset. Based on previous work, Griffin et al. [31] has postulated that performance of anomaly detection of the existing transfer-learned networks can be improved by a different architecture with an optimized recognition representation. Hence, the present study levelled at investigating the impact of an optimized recognition representations on AD accuracies by introducing some Convolutional Neural Network (CNN) architectures with varied sizes of final pooling layers.

With two different datasets, we trained several CNNs in different pooling layer sizes and tested with two different anomaly detection tasks for transfer learning. We evaluated and analyzed these results and compares them to a conventional detection method based on a traditional statistical model. The results of our experiment showed that, with our proposed optimization framework, anomaly detection performance of CNN scaled linearly with the size of feature extraction layers. Additionally, we found that transfer learning in anomaly detection task from a less varied domain to a more complicated one was infeasible to achieve results comparable to the conventional method. From our provided overhauls of data distribution, however, we noticed that our optimization architectures proved effective in such an illed domain, as CNNs still managed to learn determinable features between classes.

Code for this project can be reviewed in the following repository:

`https://github.com/pleaseRedo/MSc-Project`

Acknowledgements

I greatly thank my supervisor Dr.Lewis Griffin for giving me this golden opportunity to do this fantastic project. I also sincerely appreciate his invaluable advice in conducting experiments, designing network architecure and organizing the write-up of my report.

I would also like to thank my mom who gave my endless encouragement and care in this stressful days for completion of my report.

Contents

List of Figures

List of Tables

Acronyms

AD Anomaly Detection.

ANN Artificial Neural Network.

AUC Area Under the Curve.

BP Back Propagation.

CNN Convolutional Neural Network.

Conv layer Convolutional Layer.

DL Deep Learning.

GPU Graphical Processing Unit.

ML Machine Learning.

MLP Multilayer Perceptron.

NN Neural Network.

ReLU Rectified Linear Unit.

ROC Receiver Operating Characteristic Curve.

SVHN Street View House Number.

Chapter 1

Introduction

1.1 Anomaly detection

Anomaly detection(AD) is a task of identifying instances of nonconformity from an analyzed data set.

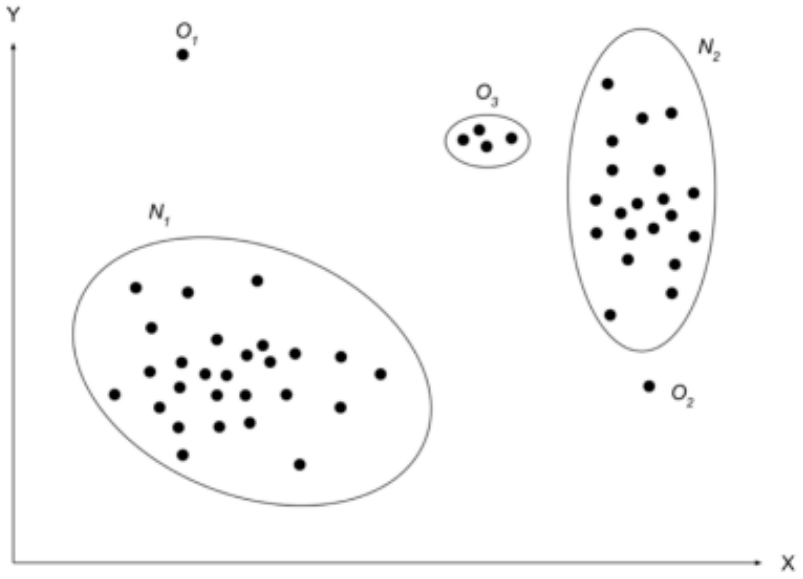


Figure 1.1: Outliers visualization [9]

Such instances are also referred to as anomalies or sometimes, outliers. As can be seen on Figure 1.1, O_1,O_2,O_3 are three outliers as they are clearly outside the clusters N_1 and N_2. Early research in anomaly detection tends to use terms *anomaly* and *outlier* interchangeably [12]. In recent works, especially in the field of computer vision, researchers seems to distinguish them where outlier detection is a part of procedure in anomaly detection [31]. Where anomaly detection is a two-step problem: data representation and outlier detection. As our project focused on representation optimization, we would, therefore, discriminate anomaly and outlier detection, where detecting outliers is a sub-task in anomaly detection.

1.1.1 Detection Methods Overview

There are many extant anomaly reviews over different domains detailing AD's comprehensive applications. From 2017 onwards, In [46], a survey of detection methods in cyber-intrusion is presented and evaluated which is an updated reviewing work published in [67] 2007. Adewumi and Akinyelu [1] gives an extensive review of AD in fraud detection. A broad study of AD techniques in medical field [51] is presented by Litjens et al. . Internet of Things and big-data anomaly detection has been reviewed by Mohammadi et al.[64]. An overview of AD in sensor networks is introduced by Ball et al. [5]. The state-of-art video surveillance anomaly detection with deep learning methods overview is brought by Kiran et al. [42].

Although in different domains, among these recent reviews, all of them mentioned the techniques of adopting deep learning in anomaly detection procedure. In [9], authors named such type of detection methods Deep Anomaly Detection(DAD).

1.1.2 Output of Anomaly Detection

One important aspect in anomaly detection in how detected anomalies are measured. Typically, those measurements can be categorized into two:

Anomaly Scores

For each data instances, anomaly score measures their degree of *outlierness*. The scoring techniques uses domain-specific threshold, also known as, decision score. One common use for scores is to rank anomalies based on those threshold proposed by data analysts, this class of output receives more attention in recent studies[78, 31]. The term *score* itself is a quite abstract concept and it is defined differently according to authors. In SVDD[78] and [31], scores are distance from points to centroids of learned model. The score could also be a reconstruction error in [32].

Anomaly Labels

Instead of giving scores, techniques in this category label each instance as either *normal* or *anomalous*. Comparing to scoring methods, label-based approach relinquishes the benefits of finding most relevant anomaly in a direct way.

1.2 Motivations

We have seen wide applications for anomaly detection in previous sections. In security imaging domain, there is a sub-branch that aims to find anomalous item over series of x-rayed security images.

X-ray imaging is mainly used for inspection. With those images, a security personnel can detect potential risks like Improvised Explosive Devices(IED) [99] and firearms [96] without the need of opening cargo or luggages.

The x-ray security operations can be broken down into four modes based on scenario: Threat Detection(TD), Semantic Analysis(SA), Manifest Verification(MA) and Anomaly Detection(AD). Since our work mostly concentrate on AD, further discussion on other three scenarios will be omitted.

Through our research, these operations are performed parallel by inspectors. For human, Such inspection task would be costly, taxing,time-consuming and error-prone [102]. These concerns necessitate an automation system which is capable of all these inspection tasks while preserving a consistent high-speed performance as well as reducing chance of corruption [75].

Although performed in parallel, only the operation TD receives most research [31] focus while achieving human-comparable results [45]. The current research stage [31] for AD are mostly exploited in satellite [73, 58, 91], less commonly in video surveillance [57] and seldomly in x-ray image [3].

To the extent of our knowledge, the first study related to specific AD on x-ray imagery is in 2013 [103], where Zheng and Elmaghraby proposed a detection method based on border-crossing vehicles x-ray scans. Although in x-ray imaging domain, the work seems developed a TD method to detect anomalies. By definition of AD, methods in [103] is not a well-defined AD, authors just happen to use the terminology 'anomaly' as an alias for 'threat' which is termed in our context of definition as discussed earlier. So Griffin et al. [31] may be credited to be the first team that conduct research about x-ray imaging AD and still remain active nowadays. The success use of transfer training domain in CNNs to accomplish anomaly detection in [31] have paved a way for us to pursuing their studies. The CNN Architecture used by them only serves as a tool for feature extraction without any research assigned to it. This network structure is designed for image recognition task rather, detecting anomalies. Therefore, most architectural design such as the final pooling layer can be optimized further to become a AD-specific network. Based on our

deep learning's knowledge, the size of pooling layer from the CNN used by [31], to be specific 1024, might not be an ideal value for anomaly detection as activations from this layer is producing very high-dimensional representations. We hypothesized that a high-dimensional representation may express good multi-class classification results but not in a two-class classification task such as AD.

Our project will seek to find empirical evidence to justify our initial hypothesis through some experiments conducted on various pooling size CNNs. Due to the fact that Network from [31] is overly large, training network with such scale might be tedious and very time-consuming. We therefore, decide to organize our study into similar structured CNN with shallower layer and less complicated image datasets.

1.3 Our Contributions

The objective of our project is to explore the potential of an optimized CNN architecture suggested by [31] for anomaly detection. concretely we will design two transfer learned anomaly detection tests to examine how well our networks performed.

The contributions of our research can be listed by two points:

1. We proposed an CNN architecture to extract an optimized representations.

2. We implemented and tested our architecture in the context of transfer learning. Specifically, we conducted extensive anomaly detection experiments on our network to determine the relationship between AD accuracy and size of representation learn layer.

3. We explored the precondition of transfer learning when applying to anomaly detection tasks.

1.4 Structure of the Report

In this section we provide an introductory background and motivations for the project. We provided an overview of anomaly detection and some related terminology. We also include a short section of how we quantified anomalous data. We then described the motivations and initial hypothesis of our project.

Chapter 2 provides preliminaries for the project briefs some technical details related to deep learning and some architectures of different networks. It also include reviews of some anomaly detection literatures. This chapter is concluded by some proposed approaches for the project. Chapter 3 details how experiments will be organized, it covers information about database, designs of network, scoring functions and evaluation metrics. Results of these experiments are given in Chapter 4. Finally, Chapter 5 provides a conclusive summary of current findings as well as our suggestions for future works.

Chapter 2

Background and related work

This chapter first reviewed approaches in the field of Anomaly detection. The first two sections were aimed at AD related literatures, these sections will provide reader with enough groundings of what AD task is. The following three sections were focused on providing technical background in terms of deep-learning, in particular convolutional neural networks some mathematical equations were displayed for clear understandings of the quantifications we adopted.

2.1 Backgournds for Anomaly Detection

2.1.1 Taxonomy of Detection Approaches

In broad view, there are three types of traditional anomaly detection mode : Supervised; unsupervised and semi-supervised. and two advanced mode: hybrid and One-class Neural Networks(OCNN) [9].

Supervised Deep Anomaly Detection (DAD) uses labelled normal and anomalous data to train a multi-classifier [11]. Thus a normal against anomaly classes predictive model is built, any upcoming unseen datum will be compared against the model to destine what

type of class it belongs to. In practice, this technique is rarely used due to its two major concerns. Meagre training data from anomalous class compare to normals'. Which in turn causes class imbalance distribution in predictive classifier [15]. Another alarming issue related to supervised DAD is the challenge in gathering representative and accurate anomaly labels [88].

Semi-supervised DAD, compare to full supervised approach, is more extensively used [9]. Methods in semi-supervised manner, abort the use of anomaly data so only normal data will be used to train the classifier. Conversely, semi-supervised training technique also applies to develope the model with anomaly data only [20]. Although several researchers [20, 19, 26]showed their effort in this special case of AD, training with anomalous set is still not commonly used[12] due to the fact of data gathering as discussed above.

Relying on the assumption of high prevalence with regrading to normal instances in test set, unsupervised anomaly detection can be trained without demanding any labelled training data, thus are more applicable [28]. several recent works including [94, 14, 95] showing the superiority [9] of adopting unsupervised learning approach in AD field by giving outperforming results when compare to traditional methods namely SVM[16],PCA[100], and Isolation Forest[52].

One big advancement in field of machine learning recently is the emergence of neural networks with their prodigious outcome over various domains [83]. Deep Learning(DL) is used to represent data hierarchically in a neural network, it achieves greater results and outperforms traditional machine learning based method given large scale of data[9]. Inevitably, recent years have witnessed the blooming of DL-based anomaly detection methods, multiple recent researches have shown that these methods outmatch the traditional ones[69, 83].

Deep hybrid and OCNN are two state-of-art anomaly detection frameworks, based on unsupervised model, both of them proposed the idea of data representation extraction

followed by a outlier detection.

In hybrid framework, deep leaning is applied to extract robust features. Those learned features are then fed into traditional classification algorithm such as Radial Basis Function(RBF) or SVM [22, 101]. To achieve stat-of-art results, hybrid methods relying on the following assumption[9]:

- A robust feature extraction mechanism, it benefits eliminating unwanted features which may camouflage the anomalies appearance.
- A robust outlier detection algorithm which can be used on high-dimensional space.

The OCNN framework uses deep learning for feature extraction jointly with one-class objective which is a hyperplane or a hypersphere for outliers separating[78]. Two 2018 experiments[78, 10] related to OCNN showed a comparable or better results than state-of-arts for complex datasets. A successful OCNN model is built upon the following assumptions[9]:

- No variation in common factors for anomaly data.

Comparing the two stat-of-art frameworks [80]: Hybrid and One-class Neural Networks. Hybrid is more scalable and tackles the "curse of dimensionality" however, it uses generic loss function so no controls over hidden layers' representation learning. In OCNN pipeline, feature extraction and outlier detection can be done simultaneously, but one significant downside of using OCNN is that, it is a highly computational intensive model which result in long-time training.

2.2 Anomaly Detection over x-rayed imagery

2.2.1 Candidate Approaches to Anomaly detection

Recall that a full anomaly detection workflow contains two parts: data representation and outlier detection. Finding outliers is a well-studied task, current outlier detection methods are based on following aspect [31]: boundaries, trees, distances and densities. In [82] an one-class SVM is proposed where a hypersphere is created and any datum outside it will be considered as anomalous. IForests [52] uses a tree-structure recursive partitioning to find isolated points where a partition threshold is the length from root to terminating node. Outliers can also be detected based on distance to k-nearest neighbours [23] or local averages [54].

Density based approach use likelihood as anomaly score. [48] uses parametric fitting to complete the density estimation, in [47], kernel-density-estimation is used for fitting parametric form.

Recent AD works[3, 4, 31] suggest that, data representation plays a more important role for AD.

One challenge for representation is that it needs to be *just right*: meagre representation fails to make anomaly atypical while too generous representation would make every data unique so does anomalies thus fails to make anomaly stands out as well. In practice, there are three representations: raw, engineered and learned.

Raw representation is least effective [31] because they usually in complex dimensions and most dimension only contributes irrelevant information.

Better than raw, engineered representations extract important features and suppress the irrelevants. However, it is known to be hard to find such representations. This situation is exacerbated in anomaly detection because there is nothing we know about anomaly class.

Among three representations, learned representations gives most preferable result. With

sufficient training data, it can achieve state-of-art performance for TD [31]. However, there is no direct equivalent approach in AD. When in the domain of x-ray imaging, it brings another challenge which is the limited number of training data. Insufficient data would make any training method infeasible.

2.2.2 Review of the State-of-arts

In previous section, we mentioned having learned representations is desirable but not directly applicable in AD. In this section, several state-of-art AD works based on learned representations will be reviewed.

In [4], an auto-encoder network is trained to represent reconstruction error of a datum. In their approach, they use x-ray images of empty and non-empty containers. In one test, they use empty one as normal,non-empty container as anomaly,then they swap the role for another test. They derived a series of features which includes hidden representation, scalar residual magnitude, the signed residual, the absolute residual and squared residual from auto-encoder. For detecting outliers, they use Radial Basis Function Support Vector Machine(RBF SVM) [81]. Based on their empirical report, the proposed work demonstrated some success in finding firearms concealed within empty cargo containers with up to 99.2% accuracy [75]. However, the proposed auto encoder method does not show comparable result when detecting non-empty cargo containers [31].

One major obstacles from [4] is the variability of items in container and the availability of training data. Transfer-learning approach is proposed to address such issue. The basic idea of transfer learning in x-ray images is to use training data which is not from x-ray scans but other types of data sets.

[3] gives a preliminary transfer-learning attempts, they transfer learnt a CNN based on a pre-trained image classification neural network called VGG [13]. Their results gives a strong indication of borrowing idea of transfer-learning for AD is viable. Authors in [3]

later follow up their study with applying transfer-learning AD in x-ray images[2]. Based on transfer learning CNN, authors use forest of random-split trees(FRST) or Isolation tree [52] which is discussed in 2.2.1 to detect outliers. In addition, authors also propose a random rotation on representation space to further improve their performance. Despite their result is just 15% above random guess accuracy, their attempts and evaluations give sufficient grounds for their most recent AD publication [31].

Approach given by [31] uses transfer learning and internal labels for training a CNN, Wolfram ImageIdentify CNN [98], to extract feature representation. Then a density method, multivariate Gaussian model is used for outlier detection.

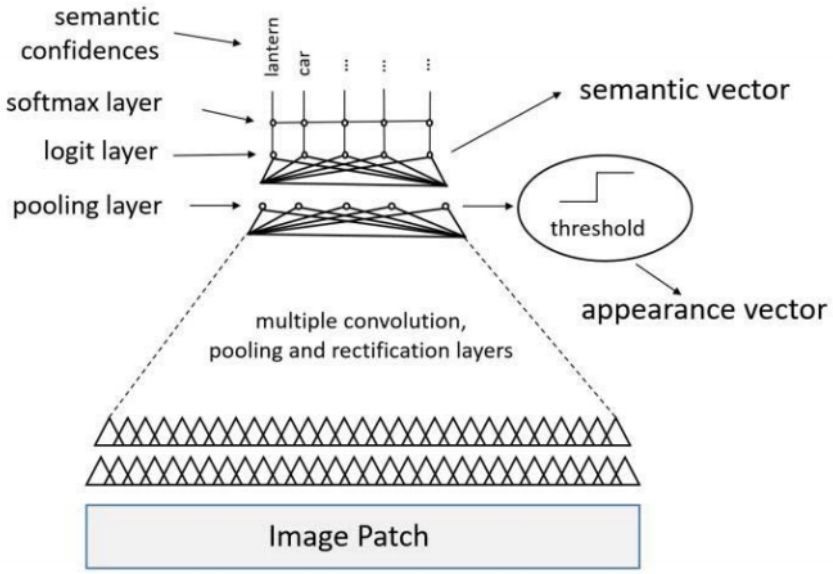


Figure 2.1: Proposed CNN architecture [31]

The CNN is trained using normal *photographic* rather than x-ray images. Figure 2.1 shows the CNN architecture used in [31], the representation is extracted based on the 1024 activations in the final pooling layer. During testing, the test set are provided by Home Office of UK government, they uses UK Stream-0f-commerce data set as normal class. A staged-threat set is adopted for anomaly class where each parcel contains normal item

with a firearm. According to their test report, the proposed AD can achieve a real-time detection speed and retains an accuracy of over 90%. However, such desirable achievements come at cost of a 18% false alarm rate which is critical problem in an security system.

2.3 Artificial Neural Network

2.3.1 Overview

Artificial Neural Network (ANN) is a mathematical approximation of biological neural system, and it consists of a collection of interconnected artificial neurons. Among these neurons, their connection allows signals to be transmitted, just like activities going on in biological synapses [24]. The origin of ANN can be traced back to 1940s, when McCulloch and Pitts [59] proposed a McCulloch-Pitts model i.e. a logical calculus expression of nervous activation. It was at the end of 1950s that the implementation [76] of the first learnable ANN named single layer *perceptron* was launched. The advent of perceptron can be regarded as a paradigm shift, yet the model have been subject to critique for its applicability which is confined to linear separation (e.g., XOR problem) by research of the following decade[62]. The limitation of previous perceptron is originated from its single-layer-architecture and shortage of effective training methods. In 1986, the introduction of back-propagation [79] and replacing single layer with multiple hidden-layers which activated by Sigmoid function remediated such limitation. Three years after, Hornik et al. and Cybenko found the *universal approximation theorem* which states network with more than three layers can approximate any functions. Albeit such advancement, neural network has failed to gain research popularity, partly due to the issue of *Vanishing Gradient* [36] and generally insufficient computing resources for training [30, p. 12].

The current resurgence in the field of ANN begins in 2006 [35], this wave of development has been further accelerated when increased amount of training data and more powerful

training hardware co-processor—Graphical Processing Unit (GPU)—has become widely available. One noticeable achievement of neural network in this particular period is the feasibility of training networks with deep-layer architecture [34, 35]

The process of assigning credits to each layer is now generally being branded as Deep Learning (DL).

2.4 Basic Neural Network

2.4.1 Background

Performance of traditional Machine Learning (ML) algorithm is, in general, determined by the choice of data representation. A good data representation can find and disentangle the *factors of variation* which defines some simplified explanations about the observed data [30, p. 4] such as the sex of human or color of a car. Acquisition of such representations in the domain of machine learning requires researchers' prior knowledge to hand craft them, this process of finding features is usually being referred as feature engineering. In practical, however, many high-level features are difficult to extract and the process of extraction itself is labor-intensive, this sometimes makes engineering work as involved as solving the original machine learning problem [30, p. 5]. Deep Learning is a desirable domain when considering leveraging such overheads in ML by introducing *representation learning*. This learning approach exploits the hierarchical structure within a complex concept such that they can be represented by simpler concepts.

The quintessential instance in DL is the *Multilayer Perceptron* (MLP). MLP is composed of many basic mathematical functions bridging the gap between input and output, many of which can be interpreted as learned representations from raw input. This input-to-output mapping is sometimes often referred to as *feedforward deep network*. Such network breaks down the data mapping over three types of layers, namely the input, hidden and

output layers, and each of the layers is formed by nodes, the fundamental units in Neural Network. Layers whose nodes are linked with all nodes from the previous layer are often named as *fully connected layers*. The function of this particular layer is to integrate previously learned features and thus allows MLP to construct complex representations in such a high-to-low level architectural way [53]. Figure 2.2 presented one basic MLP for object recognition task. Nodes depicted in Figure 2.2 are represented by circles and connections between nodes are symbolized by unidirectional arrows.

Input layer is the first layer in MLP, which is expressed as a one-dimensional vector representing raw data inputs. In the context of object recognition, as shown in Figure 2.2, the input layer is located at the bottom-most row, and the inputs are flattened RGB image pixel values. One drawback of handling image input by using flattened image vector is that it will compromise the spatial coherence or, concretely, local pixels, which, is necessary in defining a good representation [6] in representation learning.

What follows the input layer is the hidden layer, most of which are as fully connected layers. It is the place where representation learning happens. Following the flow of simple to complex learning structure, representation learned in first hidden layer is very defined and only low-level features like edges will be extracted. Complexity of learned feature increased as the depth of network going deeper, as discussed earlier, this is due to nodes from current layer are integrating all information from previous layer. Illustration of these process can be found in example in 2.2, features in lowest level are all edges, middle level feature such as corners and contours are detected in subsequent layer, the final hidden layer always gives most complex, close-to-object shape. At the end of final hidden layer, different activation functions are adopted based on how many classes reside in training set. Typically, a sigmoid is used for binary case but softmax is more appropriate for multiple classes.

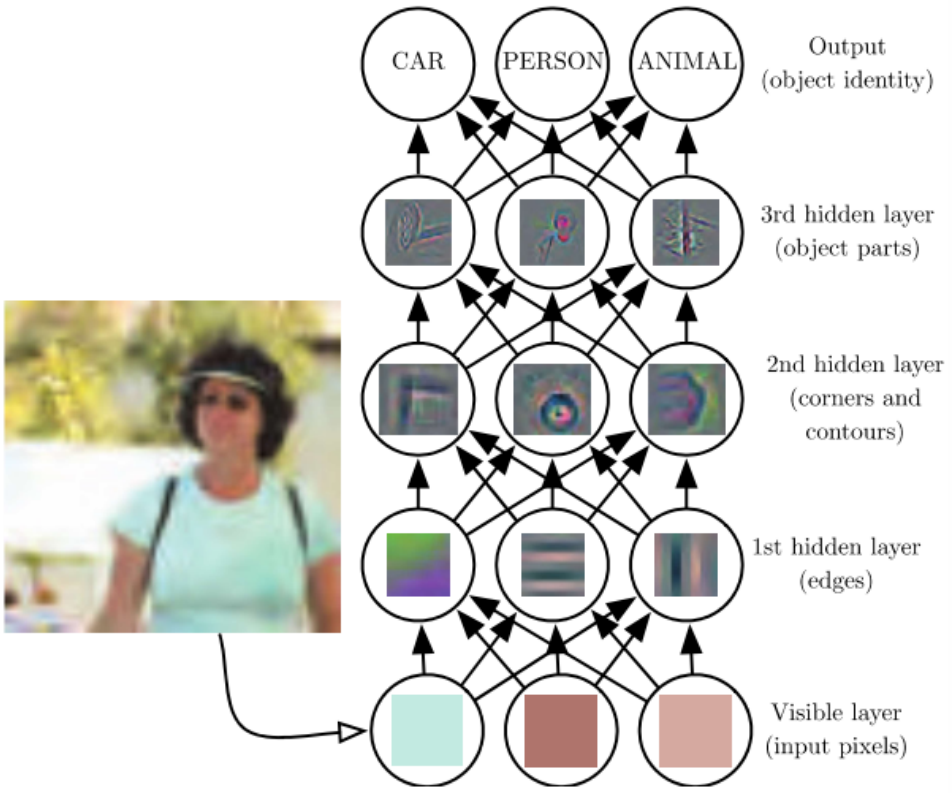


Figure 2.2: Visualization of a simple MLP architecture. Figure from [30, p. 6]

Layer followed by the final hidden layer is the output layer, which, as its name already implies, gives the output of a network. Depending on the task we are addressing, outputs could be a single node for binary classification or multiple nodes for multi-class categorization. Typical values for output layer are the beliefs of certain category, e.g., car, person and animal as in the case of Figure 2.2.

2.4.2 Forward Propagation

The process of mapping input of a network to its output through all nodes of hidden layers can be defined as *forward propagation*. Before diving into the details of neural network, let us begin with the explanation about *logistic regression* [17], as multi-layer perceptrons are somewhat related to it. In essence, we can regard a classic logistic regression as a

single-layer perceptron specialized for a binary classification where the input is directly mapped to output without any hidden layers in between.

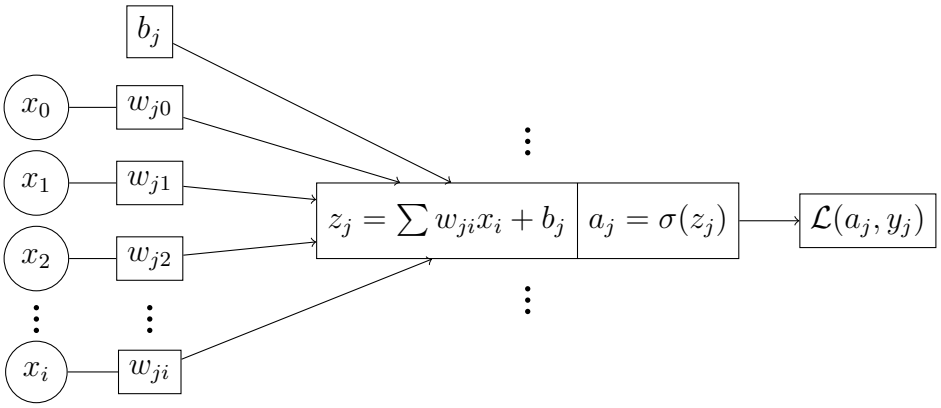


Figure 2.3: Signal flows for a node j in a MLP (with a Loss Function $\mathcal{L}(a_j, y)$ at the end, it is equivalent to a logistic regression)

Figure 2.3 presents a process of how data propagates through a node j of one layer. The input of j is denoted by a vector $\mathbf{x} \in \mathbb{R}^i$, which is given by $\mathbf{x} = \{x_0 \quad x_1 \quad \cdots \quad x_i\}$ where i stands for the number of input nodes. Each input x_i is associated with a input-node-wise weights $\mathbf{w} = \{w_0 \quad w_1 \quad \cdots \quad w_i\}$, together with bias term b, the given node first computes:

$$z = \sum w_i x_i + b$$

which can be written in the form of an affine transformation:

$$z = \mathbf{w}^T \mathbf{x} + b$$

At this stage, we need to apply non-linearity to our functions by means of activation function σ. For logistic regression, sigmoid function:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

is commonly used to map values into the range of 0 and 1 which can be directly interpreted as probability of a class in binary classification. However, sigmoid activation is obsolete in modern deep learning community with several issues, it is not zero-centred and is suffering gradient vanishing. Although the shifted version of sigmoid, tanh, addressed the former issue, problem of gradient vanishing remains unsolved. Use of Rectified Linear Unit (ReLU) [29]:

$$\sigma(z) = \max(0, z)$$

can help to solve gradient vanishing problem, thus it becomes commonly used activation functions in training deep neural networks [71]. In practice, using ReLU would introduce dead ReLU problem [97], but this problem can be resolved with Leaky ReLU function via applying a minute gradient flow. In our project, only some relatively small sized neural networks are trained and the use of Leaky ReLU did not better the model performance significantly [55]. Thus, activation of Leaky ReLU is not necessary and ReLU is adopted for our activation function through out the whole project.

In practice, making prediction using a neural network relies on the value of weights and biases, and good prediction cannot come with random initialized values thus these values have to be properly assigned first. In DL, assigning optimal weights and biases to a network is called *backward propagation* (BP).

2.4.3 Loss Functions

At the end forward propagation, we obtain a probability distribution over classes. At this point, we use *loss function*($\mathcal{L}(a_j, y)$ in Figure 2.2) to measure errors the network makes or, in other words, quantifying the closeness of network's prediction and ground-truth. Therefore, the training process (BP) is essentially a process of minimizing errors. Inputs of loss function are values from output layer and true labels, this requires both inputs are

of the same representation. Considering that most labels are just some scalars indexing the class, for binary classification, outputting single probability value is sufficient, but in terms of multi-class, we need to use one-hot encoding to map index into a vector $\mathbf{y} \in \mathbb{R}^k$ where k is the number of classes in training set. Name "one-hot" comes from the structure of encoded vector where only the index of true label is "hot"(has value 1) and 0s elsewhere.

Recall the loss function, it can be formally expressed in the following form:

$$\mathcal{L}\left(\hat{y}, y\right)$$

where $\hat{y}$ is the prediction from our network for a single data point. The function that measures average errors over the entire training set is named *cost function* and is defined as follows:

$$J(\theta) = J(\mathbf{w}, \mathbf{b}) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}\left(\hat{y}^{(i)}, y^{(i)}\right)$$

where $\mathbf{w}$ and $\mathbf{b}$ are parameter sets for weights and biases respectively, superscript i denotes i-th training example and m is the total number of examples. In some machine learning literature [7, p. 256], they may use some term called Objective Function which is a general name given to any function in optimization task. Comparing to cost function, objective function includes some penalizing or regularizing terms for better fitting performance. Discussion of regularization is beyond the scope of our project, further readings of objective function can be reviewed to [7]. Using cost function without any model complexity penalty is sufficient for our research objectives.

One simple approach for now in defining our lost function is to use square loss:

$$\mathcal{L}\left(\hat{y}, y\right) = \frac{1}{2}\left(\hat{y} - y\right)^2$$

However, such function is not applicable in our optimization approach which will be dis-

cussed in subsequent sections, due to its non-convexity in logistic regression[85], this is because the non-linear activation would break convexity. Doing so would eventually put us into the situation of dealing with multiple local optima [27]. Before defining our convex function, let us first look at the output of our network, it is a distribution, encoded label is also in a distributional representation. Thus, we can use the function from information theory called Kullback-Leibler Divergence as a metric to quantify similarity between probability distributions, this function for two discrete distributions P, Q is given by:

$$\begin{aligned} \mathrm{KL}(P\|Q) &= -\sum_{x\in\mathcal{X}} p(x)\log q(x) + \sum_{x\in\mathcal{X}} p(x)\log p(x) \\ &= \mathrm{H}(P,Q) - \mathrm{H}(P) \end{aligned} \tag{2.1}$$

where term $\mathrm{H}(P,Q)$ denotes the cross-entropy between P, Q, and term $\mathrm{H}(P)$ denotes the entropy of P. In essence, the learning process of our network is a process of making predictions $P(training)$ as close as possible to ground-truth $P(label)$, which is equivalent to minimizing KL-divergence $\mathrm{KL}(P(label)\|P(training))$. In practical, we can just compute cross-entropy term from (2.1) for optimization since entropy of ground-truth $H(label)$ is constant. Fortunately, cross-entropy is a monotonic function(convex in logistic regression), therefore, we use this function as our loss function throughout the entire experiment. Consequently, we can define our cost function as:

$$J(\mathbf{w},\mathbf{b}) = \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}\left(\hat{y}^{(i)}, y^{(i)}\right) = -\frac{1}{m}\sum_{i=1}^{m} y^{(i)}\log\hat{y}^{(i)} + \left(1-y^{(i)}\right)\log\left(1-\hat{y}^{(i)}\right) \tag{2.2}$$

2.4.4 Backward Propagation

After forward propagation, the network is measured by a cost function (2.2) which indicates how well a network's predictions are via the value of errors. The optimization objective is to let error information flows backwardly which allows parameters of network being

updated.

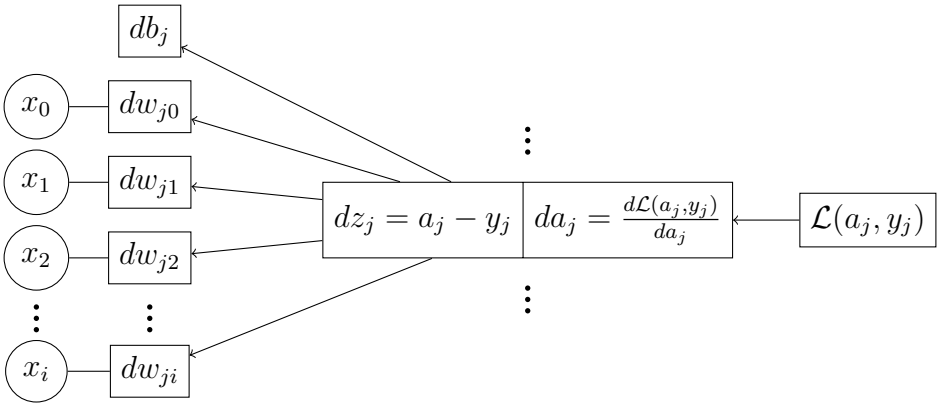


Figure 2.4: Schematic of backward propagation at layer j

One classic optimization approach in statistics is to use so called stochastic gradient descent[74, 61], with this algorithm, parameters are updated with the negative direction of gradient from loss function. Figure 2.4 depicts how parameters are updated based on gradient through propagating errors backwardly. However, such optimization requires finding analytical expressions for the whole parameters set when each data examples presented, which, in deep learning, is infeasible due to the heavy computational cost.

A better efficient way of finding gradients in neural network is termed backward propagation [79]. The central idea of BP comes fairly straightforward, the chain rule: given two functions f and g and let $y = g(x)$, the derivative of their composition $z = f(y) = f(g(x))$ can be expressed by:

$$\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx}$$

Applying chain rules to neural networks which is just a multivariate and higher-dimension cases [25], requires finding Jacobian matrix J of $f(g(x))$:

$$J_{f(g)}(x) = J_f g(x) \cdot J_g(x) \tag{2.3}$$

Function 2.3 proposed a more efficient way of finding gradient. In context of backward

propagation, it is common to use this Function 2.3 with gradient descent where the network is feeded with entire training set to compute gradient of parameters in one times. Due to the memory concern, our project uses an alternative optimization method called mini-batch gradient descent [77] where only a small portion of training data is feeded to network.

2.5 Convolutional Neural Network

2.5.1 Overview

Convolutional Neural Network (CNN or Conv Net) is an extension to deep neural network except having different layer type, both of them can be fed and trained in the same way.

2.5.2 Convolutional Layer

As discussed in section 2.4.1, in computer vision problems, classic NN discard the information from neighbor pixels [89]. In addition to this, another defects from NN is the memory-unfriendly input representation. For a RGB image with size 128 by 128, it would result in an 6 million-dimensional input vector. Reason why Convolutional Neural Network is superior [44] in vision problems than traditional deep neural network is because the introducing of a Convolutional Layer (Conv layer).

Most important computations inside a Convolutional Layer is called convolution and this is where the name *convolutional* comes from. A convolution consists of filters and outputs a feature map, filters are in relatively small size (e.g. $[3 \times 3]$ or $[5 \times 5]$) as the one in Figure 2.5. These small-scaled filters are sliding through the width and height of input feature map, computing dot products with entries they overlapped and each dot product forms an entry for output feature map.

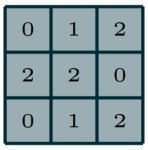


Figure 2.5: $[3 \times 3]$ filter example. Figure from [21]

An example of how a $[3 \times 3]$ filter is convolved with a $[5 \times 5]$ input feature map is given in Figure 2.6:

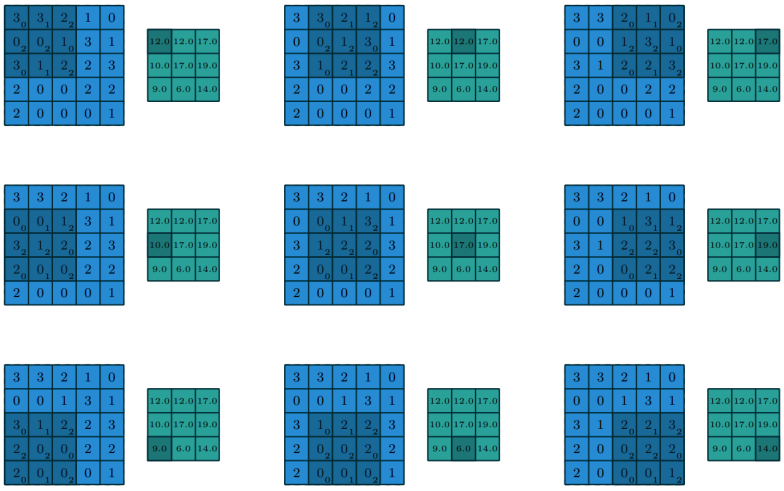


Figure 2.6: A convolution example with filter size $[3 \times 3]$ and input feature map size $[5 \times 5]$. Figure from [21]

Above example depicts how convolution works in 2-d, this operation is expandable to higher-dimension. In most CNN architecture, input image might be of 3-dimension, to comply with 3-d input(or input feature map if previous layer is not the input layer), cubic filters are needed for convolution. These filters will sliding within the volume of input doing dot product and still output a single $[1 \times 1]$ entries. Sometimes, multiple number of filters might be used to extract more features, once all filters are iterated, the collection of all those corresponding feature map will form a cubic output volume, the depth of volume is given by the number of filters used.

For a Convolutional Layer, several hyperparameters are used for tuning, what we have just covered is the *filter counts*. *Stride* is another parameter used to control how many

pixel skips a filter must take before making next dot product. The third which is the last hyperparameter in Convolutional Layer is called *padding*, it is used to maintain the input and output size consistency by adding a zero-valued border to the input volume. Further details of hyperparameter tunning will be given in the chapter of **Implementation**.

2.5.3 Convolutional Layer Frameworks

One decisive factor that differentiates CNN from classical Artificial Neural Network is its convolutional layers. Besides the aforementioned Conv Layer, classical CNN architecture consists of two additional types of layers, namely pooling (downsampling)layer and fully-connected layer. Purpose of using pooling layer is to reduce resolution of feature maps, this would enables spatial invariance to distortion and translation [72, 49]. Mainstream pooling layers have two variations: max pool and average pool.

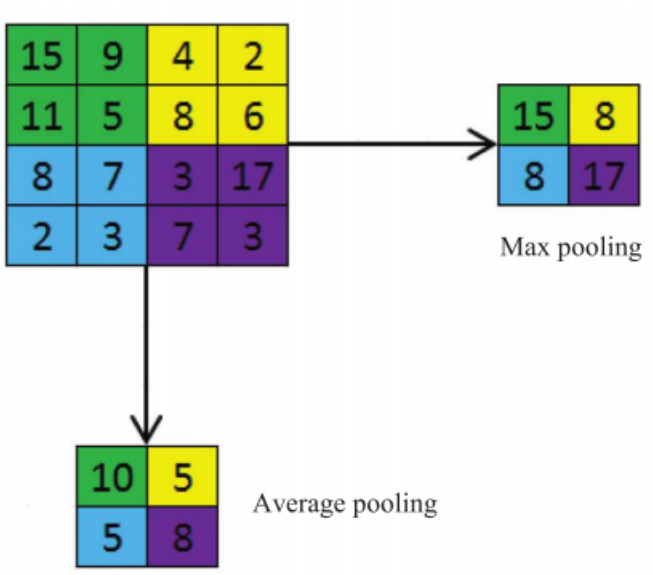


Figure 2.7: Two types of pooling layer example. Figure from [72]

The computation of both pooling is fairly simple, Figure 2.7 demonstrates how these two layers is operated on a $[4 \times 4]$ feature map. We can see the original map is downsampled to

$[2 \times 2]$, entries with same color are pooled to become one entry. The difference between two pooling method is based on how entries are pooled (i.e., averaging or extracting maximum). Generally, pooling layers are always come right after a Conv layer. In practice, pooling layer can be replaced by a series of convolutional layers with properly chosen stride and filter size. This does not cause any loss to accuracy but rather, performs slightly better when training for small sized CNN [86, 38]. Hence, for our project, we will avoid using of pooling layer but employing convolutional layer instead.

Before output layer, there will be one or multiple fully-connected layers as described earlier to connect the learned representation from final pooling layer or equivalent Conv layer to the final output layer.

Modern literature in deep learning CNN may also include some regularization layers such as dropout [87] and batch normalization [39]. Having these layers can effectively prevent model overfitting, therefore, we will consider include them into our CNN architecture.

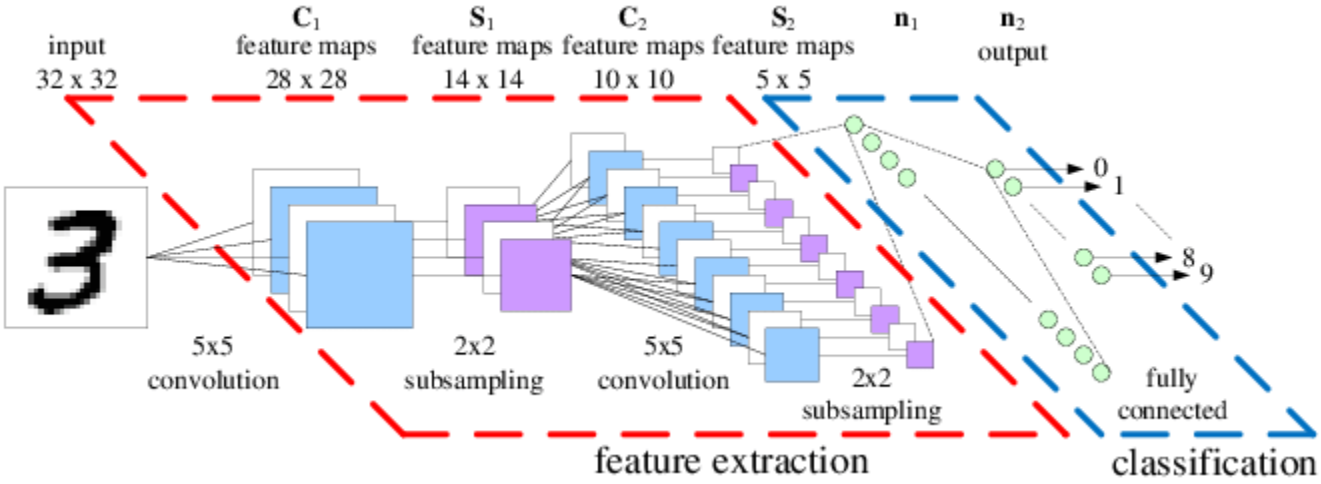


Figure 2.8: CNN architecture for hand-written digit classification data set. Figure from [68]

CNN architecture from Figure 2.8 have demonstrated how an image of a hand-written digit was classified. As we can see, the size of feature map decreased through propagation but increased in terms of the amount of them. Once CNN completes feature extraction,

all feature maps will be flattened and connects to a fully connected layer, similar to a traditional network, outputs from fully connected layer are activated by a softmax function for classifying input digits.

Recent CNNs are deeper and having more complex blocks in architecture for state-of-art computer vision performance such as VGG-16 [84] and Inception network [90]. Using network like these may grant us better detection result but the purpose of our research does not focus on performance but some results generated by running many different CNNs. Training multiple networks with architecture like VGG-16 or Inception may cost months of efforts which seems rather impractical for our project given the limited amount of time. Since one of our training set consists hand-written digit which is the same set used in 2.8, therefore, our proposed CNN structure is highly inspired by this architecture. Based on network depicted by 2.8, our network also added dropout and batch normalization layer and using convolutional layer for subsampling. Finally, the choice is learning rate is determined by using Adam Optimizer[41], additionally, a annealer scheduler [43] is include to prevent optimizer running into local minima issues. Details of network structure and all hyperparameters can be found in Table 2.1

Table 2.1: Proposed CNN architecure

Layer Types	Kernel Size	Stride	Units	Output Map Size	Batch Norm	Activation	Paddings
Input	None	None	1	28×28	False	None	False
Convolution	3×3	1	32	26×26	True	ReLu	False
Convolution	3×3	1	32	24×24	True	Relu	False
Convolution	5×5	2	32	12×12	True	Relu	True
Dropout Layer(0.4)							
Convolution	3×3	1	64	10×10	True	Relu	False
Convolution	3×3	1	64	8×8	True	Relu	False
Convolution	5×5	2	64	4×4	True	Relu	True
Dropout Layer(0.4)							
FlattenLayer( )							
Fully Connected	None	None	Variable	1×1	Yes	Softmax	False
Hyperparameters							
Epochs	20						
Optimizer	Adam	$\alpha = 0.001$	$\beta_1 = 0.9$	$\beta_2 = 0.999$	$\epsilon = 1e^{-08}$		
Loss	Cross-Entropy						
Annealer	$\lambda(x) : 0.001 \times 0.95^{(x+epochs)}$						

Chapter 3

Experiments

Our experiments are based on the following hardwares: 16GB RAM, GTX1070 Max-Q GPU and Intel Core i7-8750H CPU.

This chapter aims to present the experiment details in our project. The proposed functions and architecture of CNNs are explained in this chapter.

3.1 Dataset

Our experiment would use two 10-class-digit data sets: Street View House Number (SVHN) [65] and MNIST [50]. Both of them will be used for training CNN and detecting anomalies. MNIST is a database for hand-written digits with a total of 70,000 examples including 60,000 examples for training and 10,000 examples for testing. SVHN data set consists house number digits obtained from Google Stree View, this database has 73,257 training data and 26,032 testing images. From training sets, 33% of data are used to form validation sets.

Images in MNIST dataset are of size $[28 \times 28]$ in grey scale, examples from SVHN are RGB images with shape of $[32 \times 32]$. *Preprocessing* techniques is applied to SVHN data,

these images are resized to $[28 \times 28]$ and converted to grey scales. Apart from keeping dimension in accordance with MNIST, all examples are normalized to map pixel value from $[0, 255]$ to the range $[0, 1]$.

3.2 Implementation

3.2.1 Proposed CNN Architecture

Our final CNN architecture throughout the experiment are given in Figure 3.1. As discussed earlier in Section 2.5.3, downsampling layer are replaced by strided convolution layers.

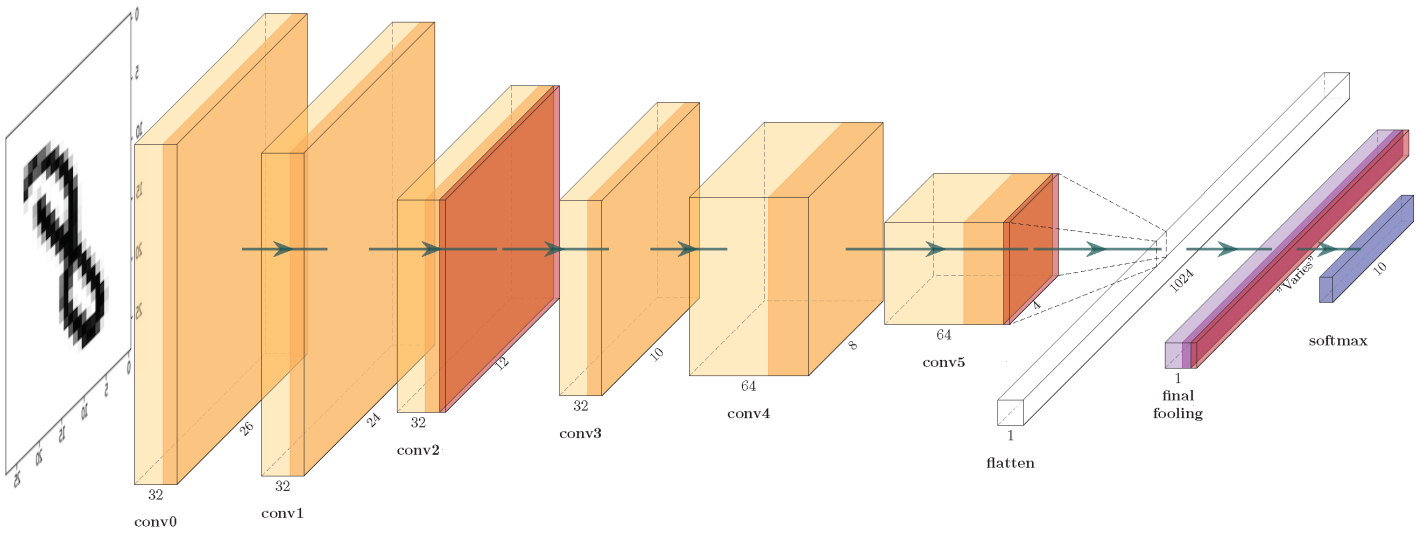


Figure 3.1: Our proposed CNN architecture. Input to this architecture is a $[28 \times 28]$ image example and it outputs a $[1 \times 10]$ "one-hot" encoded lable. This network consists six Conv layers colored in champagne, without padding, the input feature map will keep downsampling which results in a 1024 parameters flattened layer. A fully-connected layer(in light violet) is added to map output from the flattened layer.

Every layer in our architecture is followed by a batch normalization layers which is highlighted in a darken color. Furthermore, layers in orange are dropout layer attached

after some batch-norm layers. The only fully-connected layer(in light violet) in our CNN are termed as the *final pooling layer* based on the naming fashion used in [31]. Our anomaly detection research is based on the learned representation from this particular pooling layer in various sizes.

3.3 Anomaly Detection Pipeline

Learning Representations

Our anomaly detection workflow can be summarized into two steps: feature extraction and outlier detection. We designed a network pool with varying final pooling layer size $[2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384]$, choice of these value is based on power-of-two heuristic[63] in order to achieve optimal performance in representational learnings[40]. We trained our networks from the pool with both datasets, later, we will retrieve values reside in those final pooling layers as the representation.

Anomaly Score Function

Our approach for detecting anomalies is to find so-called anomaly score for each data representation, the higher the value, the higher in chance an example is an anomaly. For the following experiments, we use the logarithm of squared *Mahalanobis distance* [60] as a metric for our anomaly score. The distance equation of an observation $\mathbf{x} \in \mathbb{R}^n : (x_1, x_2, x_3, \ldots, x_n)^T$ with means $\mu \in \mathbb{R}^n : (\mu_1, \mu_2, \mu_3, \ldots, \mu_n)^T$ is given by:

$$Distance_M(\mathbf{x}) = \sqrt{(\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu)} \tag{3.1}$$

where $\Sigma \in \mathbb{R}^{n \times n}$ is the covariance matrix for observation $\mathbf{x}$. Parameters in Equation (3.1) are obtained through fitting a *Multivariate Gaussian Distribution* [93, p. 6], the distribution

for a variable $\mathbf{x}$ is :

$$p(x;\mu,\Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right\} \tag{3.2}$$

The rationale of using Gaussians for modeling anomaly-normal data distribution is because the central limit theorem and data which is not normally distributed can also be transformed into a Gaussian [92].

Note, our equation involves inverting large matrices, to eliminate the potential of inverting a singlar-matrix, we added an identify matrix $\mathbf{I}$ scaled by ϵ to Σ. Choice of ϵ values are based on finding ϵ through multiple runs with various values, so that having an optimal anomaly detection result.

Dataset Manipulation

When in the stage of outlier detection, we first divide data set into two-class: anomaly and normal. These sets are organized as follows, for each class from 0 to 9, assign examples with current label as anomaly and the rest with normal, thus we get 10 different anomaly-normal data sets. Among the normal sets, we further split them into two folds:*sample* and *test-normal*. Then, we use data from sample set to fit 3.2. Examples from anomaly and test-normal set are mixed to form a final *mixed test* set which are used for evaluating AD performance.

Evaluation Method

Adjusting the threshold value for determining an anomaly will produce statistics into four sets for each threshold: *true positive*, *true negative*, *false positive* and *false negative*. Based on these stats, we can form an evaluation curve called Receiver Operating Characteristic Curve (ROC) which is based on the calculated true positive rate against false negative

rate. This curve visualized a binary classifier's diagnostic ability by varying its discriminating threshold. This metric is highly suitable for our experiments because it properly handles the continuous score thresholding values and it also fits class-imbalance situation [8]. Area Under the Curve (AUC) is often computed once we obtained ROC, it computes the integration of the curve, thus ranges between 0 and 1. Value of AUC being 0.5 stands for machine is doing complete random guess where 1.0 indicates a perfect classification.

3.3.1 Summary

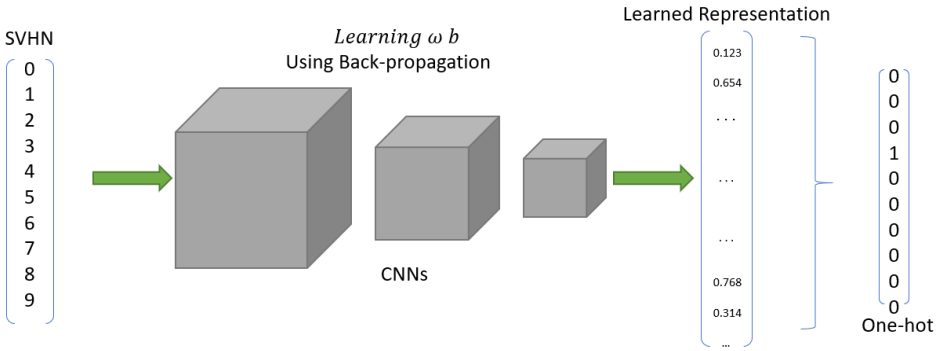
Having explained the prerequisites of our experiment, here are how our experiments are conducted, we begin with training multiple CNNs from pool, feed CNNs with all data sets; *sample*,*mixed test* to get representations. Representations from *sample* set are used to fit Equation 3.2, this will model the distribution of normal set cluster. Detecting whether a given example is an anomaly or not are based on how further it is towards the normal data cluster, therefore, distances for every examples from *mixed test* are measured by Equation (3.1). Finally, ROC graphs for each networks over 10 anomaly classes are presented and the associated AUC is computed.

One of our research is to explore idea of applying transfer learning in AD, therefore, dataset used for training CNN is different to the one used in outlier detection stage. In practice, our first experiment namely SVHN experiment uses SVHN for training network and uses MINST for detecting anomalous data. Conversely, we also conduct the MNIST experiment where we use MNIST for training CNNs and SVHN for AD.

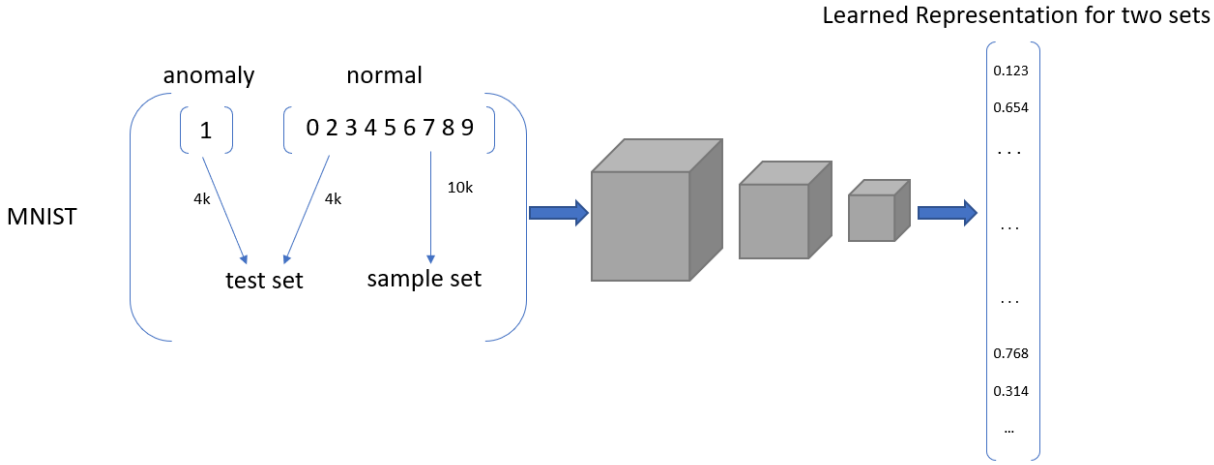
At the end of each experiments, a baseline AD algorithm based on fitting Multivariate Gaussian directly to raw pixel data without any manipulations. This baseline is set to signify the feasibility of our proposed transfer learning AD framework.

Figure 3.2 depicts our overall experiment pipeline for SVHN-experiment, this pipeline is similar to what we have done in mnist experiment with only data set being swapped.

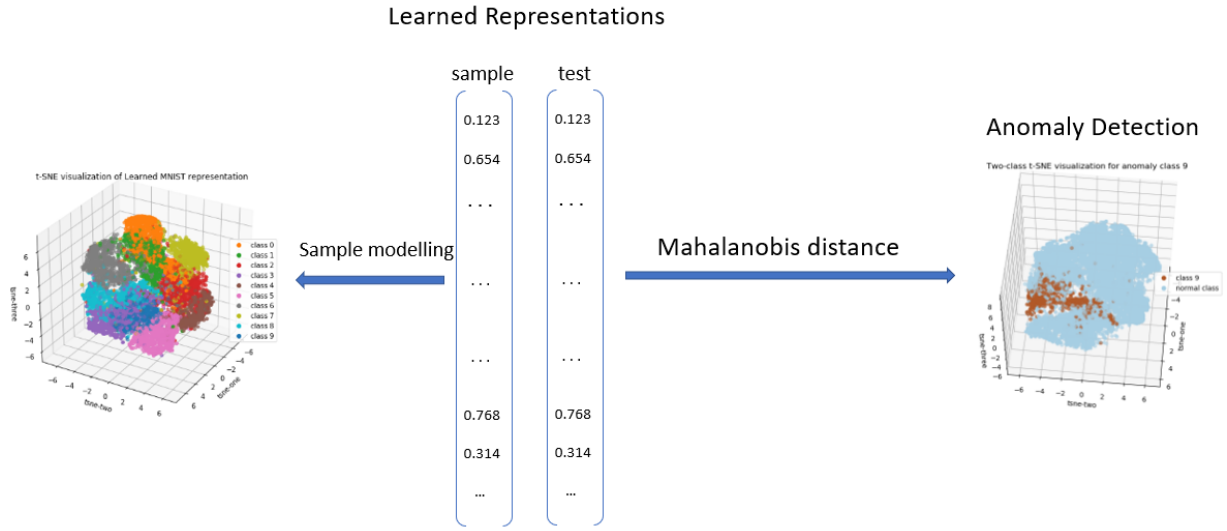
From the figure, we only gives one anomaly-normal splits. In practical, we will repeat this process 10 times because we have 10 anomaly-normal pairs.



(a) We use data from SVHN set to train our CNNs with different pooling layer size.



(b) We split MNIST into anomaly and normal set, based on that, we form the test set and sample set. And use learned CNN to extract their representations.



(c) We use data from sample set to fit distribution of normal examples. Then we use our scoring function to measure the distance of examples in test set to centroid of normals. Larger anomaly score indicates more likely that data is an anomaly.

Figure 3.2: Schematics of our experiment.

Chapter 4

Results and Analysis

This chapter will demonstrate our results and analysis of our project's experiment. This chapter also provide the process of gathering some hyperparameters such as ϵ and layer size.

4.1 Determing Epsilon

Before beginning anomaly detection, we need to specify our values of ϵ first, what we have done is to varing ϵ from the set:

$[0, 0.5, 0.1, 0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001, 0.00005, 0.00001, 0.000005, 0.000001]$

We tested our networks from 3 network pools and find out, as given in Figure 4.1, that networks with ϵ value of 0.001 will reach their optimal performance in anomaly detection except few rare cases where covariance matrix Σ is invertible already. Based on this findings, we will keep $\epsilon = 0.001$ throughout the rest of our experiments.

We did the same things for our baseline algorithm. The finalized value of $\epsilon_{baseline}$ is 0.1.

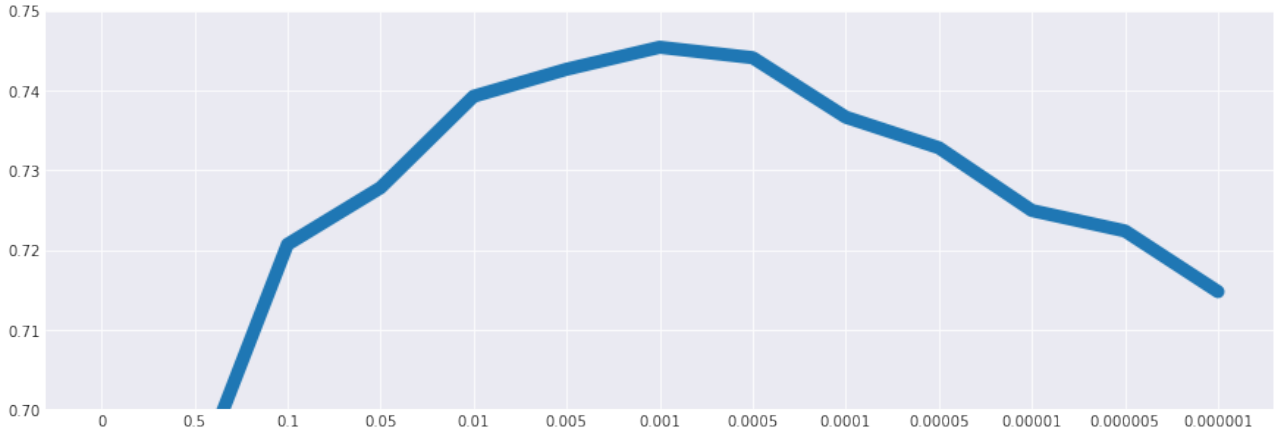


Figure 4.1: Results from one of our epsilon test.

4.2 SVHN Experiment

One of our research objective is to examine how size of final pooling layer affects AD performance over transfer-learned representation. For this purpose, we designed a pool of 14 CNNs with different final pooling layer sizes ranges from 2^1 up to 2^{14}. Practically, training a neural network begin with a random initialization phase, which means two networks even with same network structure may still give varied results. To get a more consolidated conclusion, under limited amount of experiment time, we additionally trained extra 6 network pools. Thus, our conclusion are drawn from the average of in total of 7 network pools with each have 14 CNNs.

During training, we keep track of their digits classification accuracies from validation set. All training accuracies data curated are somewhat showing similar tendency as the one given in Figure 4.2.

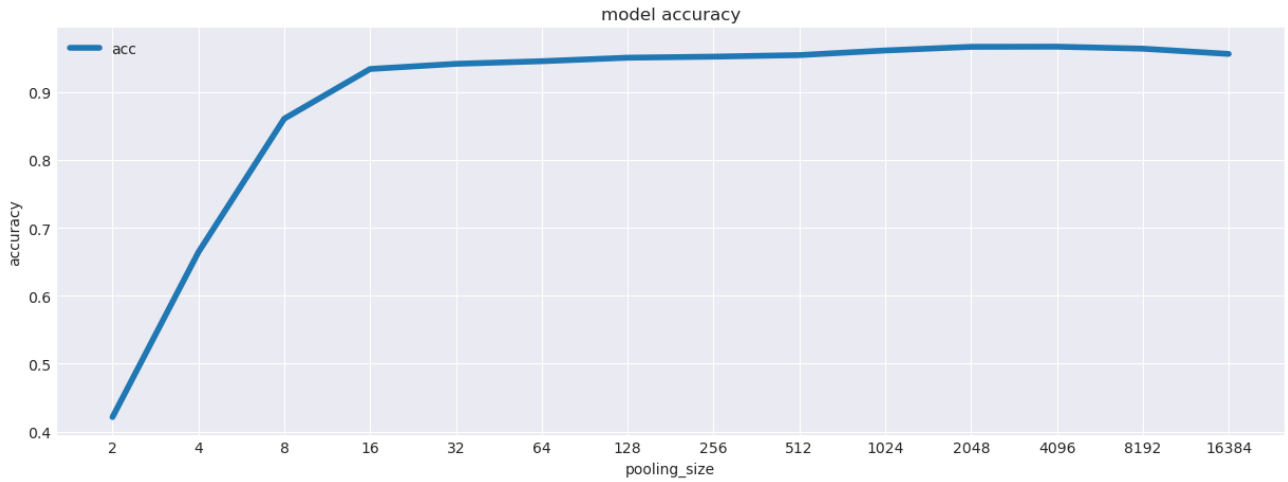


Figure 4.2: CNN Validation accuracy from one of our trained network pool.

We can see having pooling size of 2 is impossible for classification task, accuracy grows with size and plateaued at size 16.

All networks are tested using test set discussed in Section 3.1, we hereby provide testing results for 4 network pools in Figure 4.3.

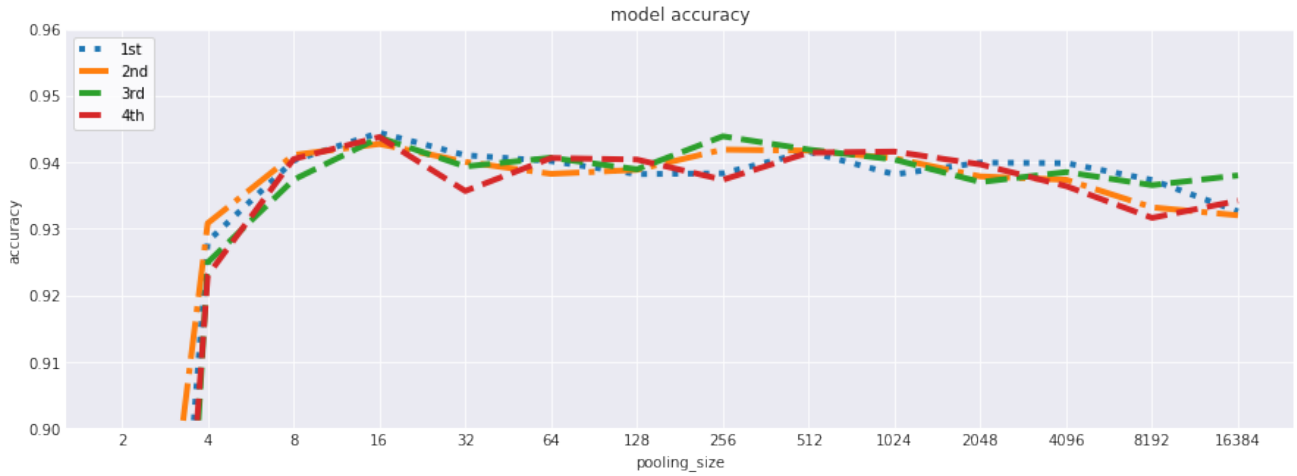


Figure 4.3: CNN Testing accuracy from four of our trained network pool. According to Validation result, accuracy in size 2 is trivial, thus we further zoomed our axes into the range $[0.9, 1.0]$

Results in Figure 4.3 showed similar accuracy curve to our validation curve with an

increased accuracy in size 4. Combine two figures together, we see our training process are giving decent results without any issue of having high variance or bias. Therefore, we conclude our convolutional neural networks are capable to learn some resultful representations from digits. With these results, we are able to proceed our experiment to outlier detection.

4.2.1 Detecting Anomalies

After experiments from previous section, we have trained 7 network pools using SVHN dataset. In this section, we will use the representations from the output of final pooling layer to fit the gaussian model and detecting anomalies.

We feed all networks with MNIST testing dataset, for each network, we extract values from the final pooling layer. Among these representations, based on their associated labels, we form the anomaly and normal sets in the form given by Figure 4.4:

Normal Class	Anomaly Class
[1,2,3,4,5,6,7,8,9]	[0]
[0,2,3,4,5,6,7,8,9]	[1]
[0,1,3,4,5,6,7,8,9]	[2]
[0,1,2,4,5,6,7,8,9]	[3]
[0,1,2,3,5,6,7,8,9]	[4]
[0,1,2,3,4,6,7,8,9]	[5]
[0,1,2,3,4,5,7,8,9]	[6]
[0,1,2,3,4,5,6,8,9]	[7]
[0,1,2,3,4,5,6,7,9]	[8]
[0,1,2,3,4,5,6,7,8]	[9]

Figure 4.4: Splits of data representations

For each data in Normal Class, we use 10,000 of them to fit distribution in (3.2) and another 4000 data are randomly picked to form *test-normal* set. Data from *test-normal* are combined with 4000 data from anomaly class to form the *mixed-test* set. Main rea-

son of only use part of data set is because of the limited computing resources such as memory. Apart from the issue of meagre memory, our experiment also was hinderd by inverting large-size covariance matrix not because the non-invertibility problem but the huge amount of elapsed-time for CPU to complete those inversions. We therefore uses CUDA programming(Python's CuPy library[66]) to speedup heavy numerical computations.

We use our proposed anomaly score function to compute scores for all examples in *mixed-test*. ROCs for each network are obtained based on the threshold between the extrema of anomaly scores. Figure 4.5 is the averaged AUC over 7 network pools for each anomalous digits over 14 different pooling layer sized CNNs.

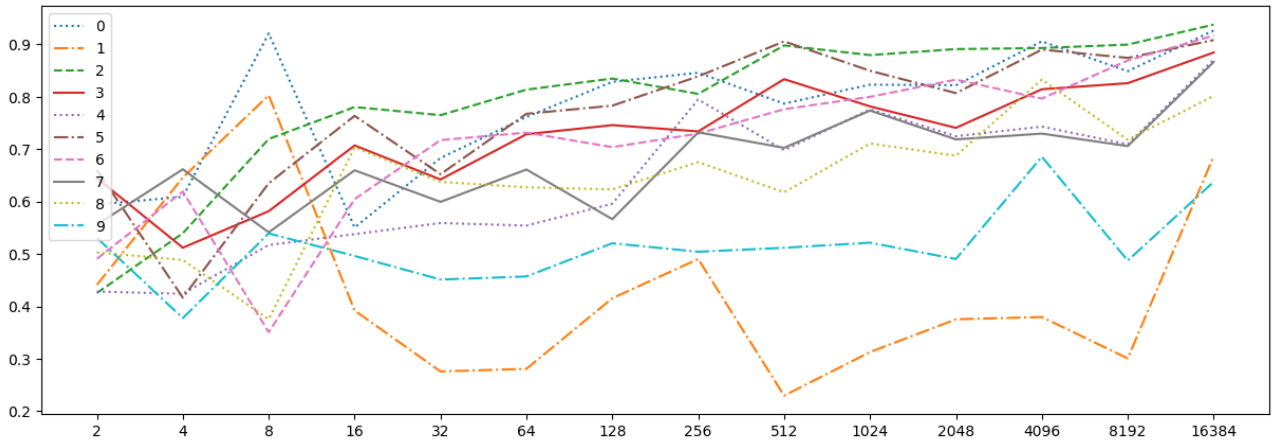


Figure 4.5: Averaged AD AUC for 10 anomaly classes for 14 networks

We can see the performance of anomaly detection for most digits are increased as size of pooing layer grows except digit class 1 and 9. The reason why detecting anomaly digit 1 is unpredictable might be because the shape of one is overly simple(shape of an edge) and having big pooling size might force the network shatterring such simple feature structure into multiple pieces and these networks cannot simply reconstruct those pieces back to edge-like feature. Class 9 shows a steady AD result with the chance close to random guess, reason behind this finding, possibly, due the learned representation of 9 does not have distinct characteristics to other classes so that the anomaly score for class 9 and normal

classes are entangled together.

Results from Figure 4.5 can be further averaged, as shown in Figure 4.6, to form an overall AD AUC for 14 different pooling size networks.

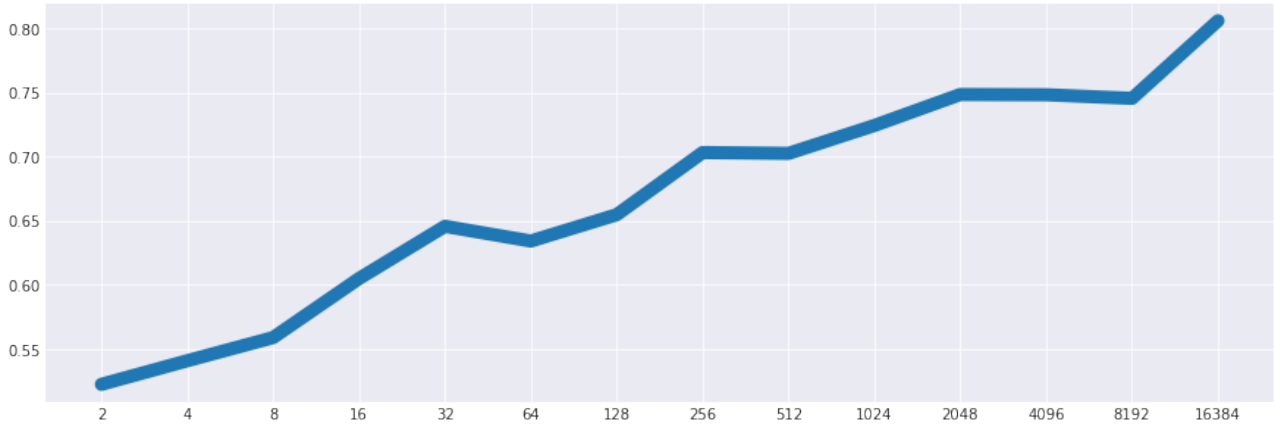


Figure 4.6: Overall AD AUC for each network

We can see the plot from Figure 4.6 showing a upward tendency in terms of anomaly detection performance. Just for the reference, our baseline method has AD performance, on average, 0.673, which means the baseline algorithm is compatible to layer size less than 128. This graph is not in the shape of what we initially expected: having overly large pooling layer size will result in a decrement in anomaly detection performance.

4.3 MNIST Experiment

This experiment are conducted in the same way as in SVHN experiment with same network architecture setup except we swap the dataset used for training CNN and for anomaly detection. Figure 4.7 showed our testing accuracy, comparing to SVHN, images in MNIST have less variational form in terms of appearance so changing pooling size would not have too much impacting on testing.

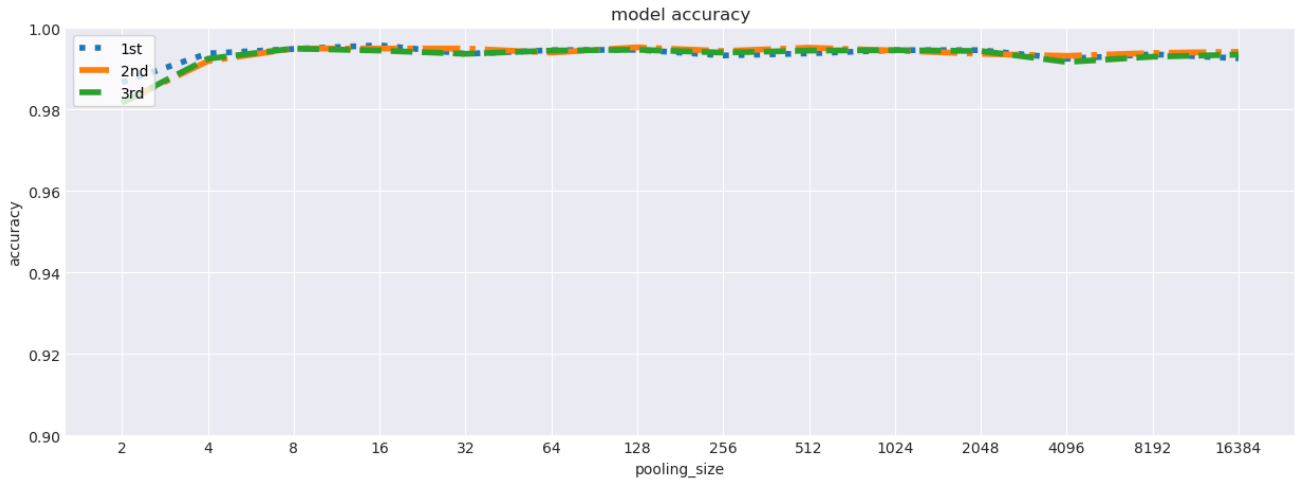


Figure 4.7: CNN testing accuracy on MNIST dataset for three runs.

The anomaly digits' class-wise detection AUC are given in Figure 4.8

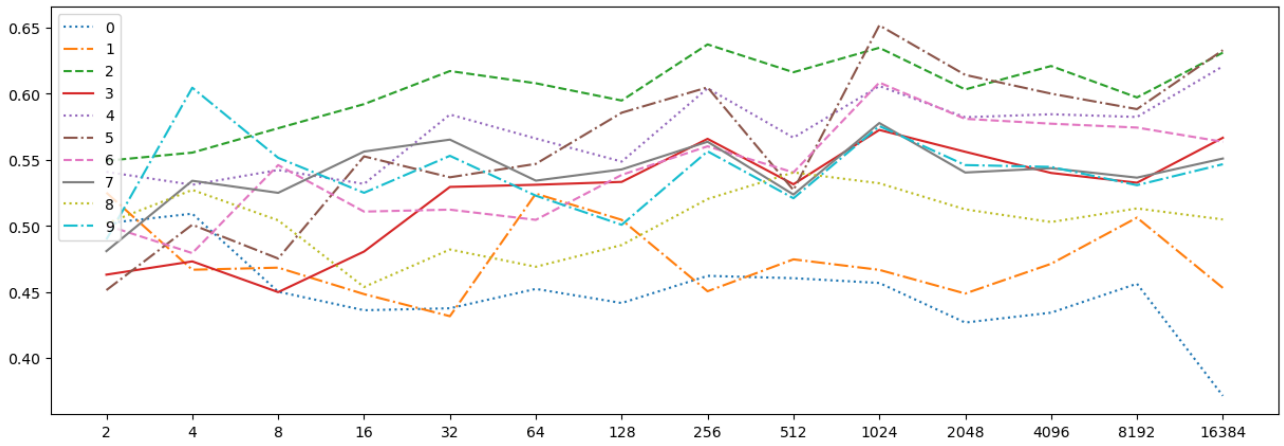


Figure 4.8: Averaged AD AUC for 10 anomaly classes for 14 networks detecting anomaly digits from SVHN dataset.

According to this result, we also observed the growing performance when we use larger pooling size network. However, class 0 and 1 are having AUC that is below 0.5, our scoring function for the given representation is unable to produce large score for these two classes. Similar to SVHN experiment, averaging values from Figure 4.8 will produce plots for overall AD performance.

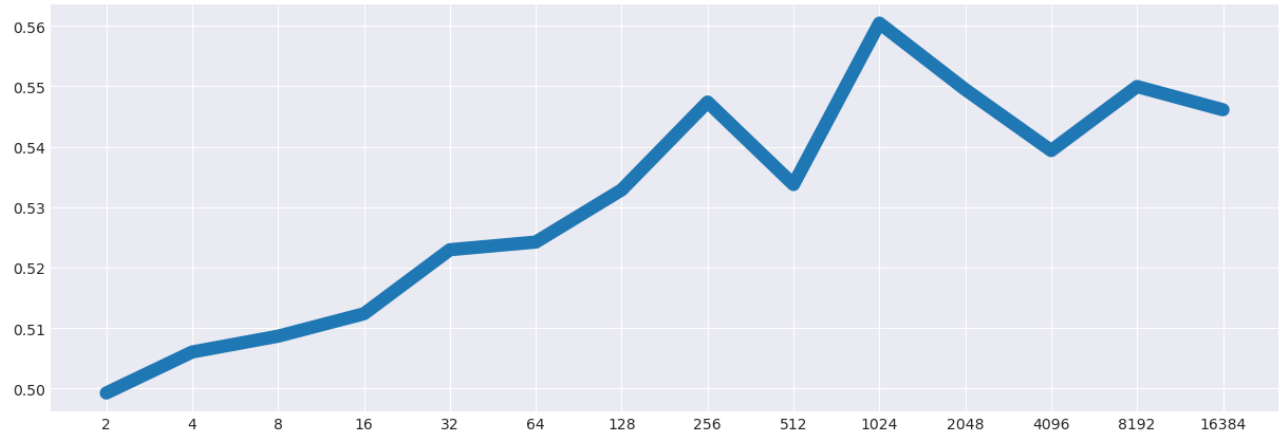


Figure 4.9: Overall AD AUC for each network when detecting anomalies from SVHN dataset.

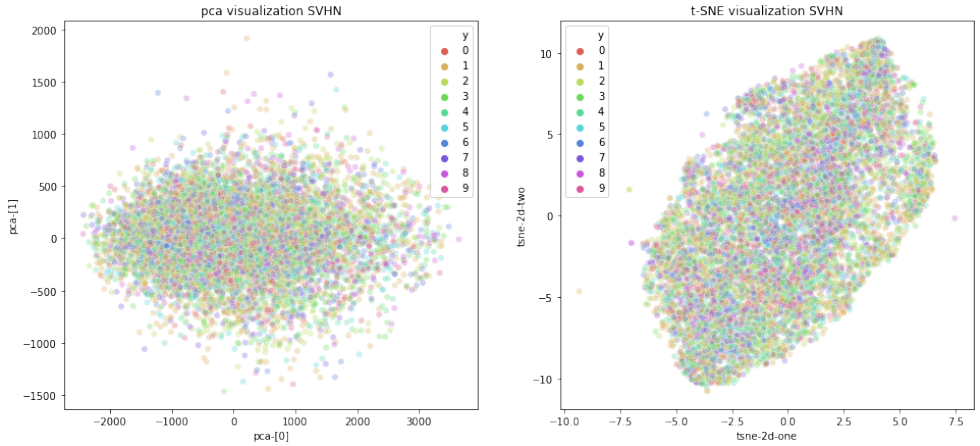
Figure 4.9 depicts the overall AD performance of 14 different network trained on MNIST. For the reference, our baseline method's performance achieved 0.508(result can be found in AppendixA) which is really means baseline for separating anomalies from SVHN is inappropriate. Although these network presented very limited anomaly detection ability, we can conclude that in the domain of anomaly detection, if one needs transfer learning, we suggest transfer from a data whose distribution comes from a larger and more varied domain.

4.4 Qualitative Inspection of Data Distributions

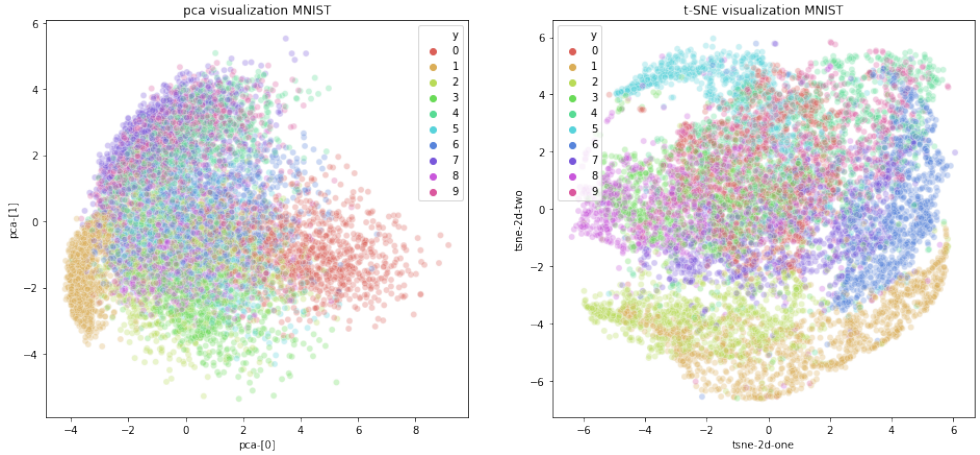
Our findings yielded by anomaly detection experiment still remain elusive, especially, the unpredictable detection rate from anomaly classes of $[1, 9]$ in SVHN experiment. Although we have provided our initial speculations for these unexpected observations, to further understand what factors contribute to such counter-intuitive result, we inspect digits' raw pixel distributions as well as the distributions of their representations. Additionally, we also inspect the anomaly scores' distribution for examples in *mixed-test*.

Firstly, we plot pixel value distributions for images from both SVHN and MNIST

dataset based on PCA dimensionality reduction [100] and t-SNE [56].



(a) Visualization of pixel values from SVHN



(b) Visualization of pixel values from MNIST

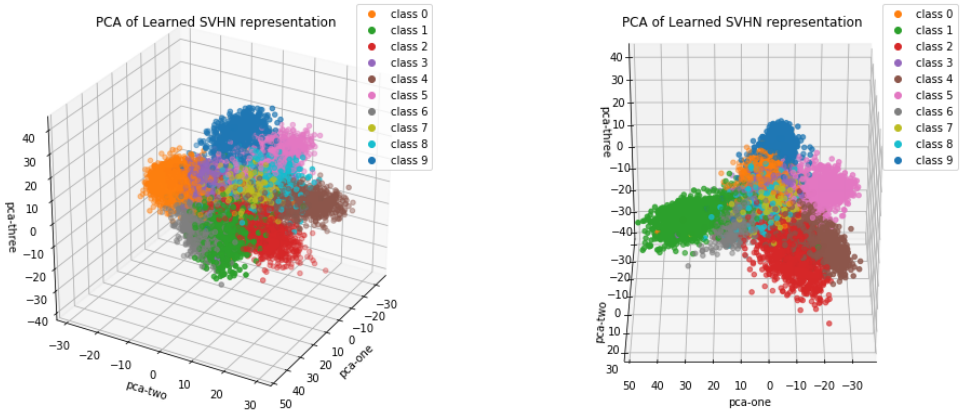
Figure 4.10: Raw pixel data visualization for our datasets.

Figure 4.10 visualized how image pixels are distributed over the space, these graphs may help to clarify why our baseline algorithm is not applicable to SVHN data(non-gaussian) while giving much decent result from MNIST. These scatter plots also evidenced our interpretation that SVHN is a more challengeable set thus, more varied domain than MNIST.

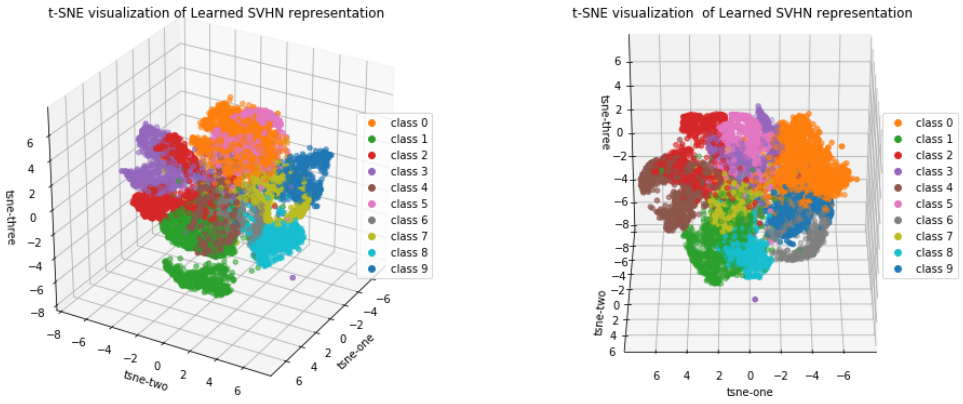
In order to visualize learned representation, we extract features from a CNN with 2048 pooling layer size because ,according to Figure 4.6, this network have shown a good representational learning ability supported by its nice AD results and layers with size

greater than 2048 do not show any significant improvements in anomaly detection. We also provide visualizations of representation's distribution from different scaled CNNs in the Appendix A, these might give reader and us good intuition of how CNNs learned representation progressively.

Figure 4.11 have justified our statement about strong feature extraction ability for our 2048-sized pooling layer CNN. Comparing to result from Figure 4.10a, this time, we can observe clear boundaries from class to class.



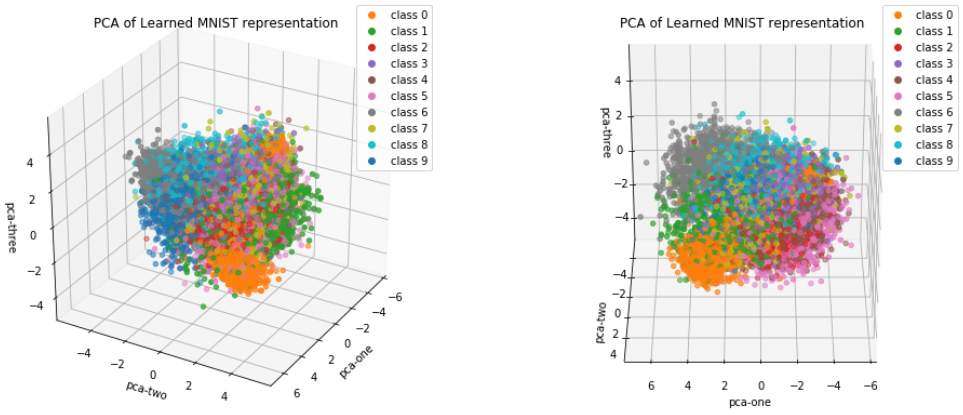
(a) Data plotted by PCA



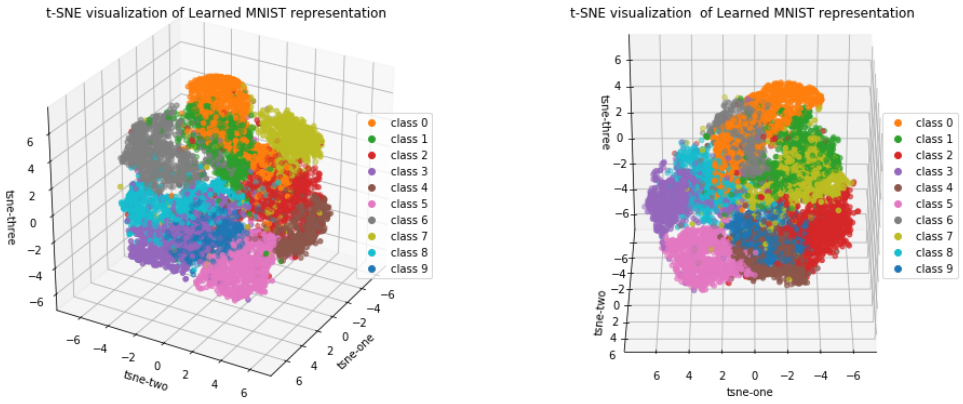
(b) t-SNE visualization

Figure 4.11: 3-D data visualization for SVHN representations learned by CNNs trained on SVHN

For SVHN experiment, what we really interested at is how our networks extract fea-



(a) Data plotted by PCA



(b) t-SNE visualization

Figure 4.12: 3-D data visualization for MNIST representations learned by CNNs trained on SVHN

tures from transferred domain: from SVHN to MNIST. Figure 4.12 illustrated the learned representation from our CNNs for transfer learning. Results from this figure shows clusters for classes from MNIST feature in normal distribution. This might give us a good intuition of why our proposed anomaly score Function (3.1) works.

Finally, we convert those plots based on Figure 4.12 to a two-class classification: anomaly and normal. Hopefully, with these new plots in Figure 4.14, the confounded detection result of class 1 and 9 can be elucidated.

In addition two class 1 and 9, we also plot class 2(anomaly class with the highest AUC)

as a contrast. From the Figures A.4a and A.4b, their distributions from t-SNE have one common characteristic, that is, they spread more widely than the distribution from class 2. This distributional presentation might partly due to high-level feature from most digits integrates simple shape like edge, digit 1 coincides to resemble in appearance of an edge and such feature are also shared by most of other digits.

Since our scoring function is measuring distance to the centroid of normal-class, ideally, examples from an anomaly class ought to be further away to the cluster formed by normal-class data with a small standard deviation from their mean. Class with an overly diffused distribution would consequently raise many false alarms due to part of its data distributed being so close to the centroid of normal's. When in juxtaposed with class 2, the manifestations of class layout from digit 1 and 9 give some corroborative data to this explanation.

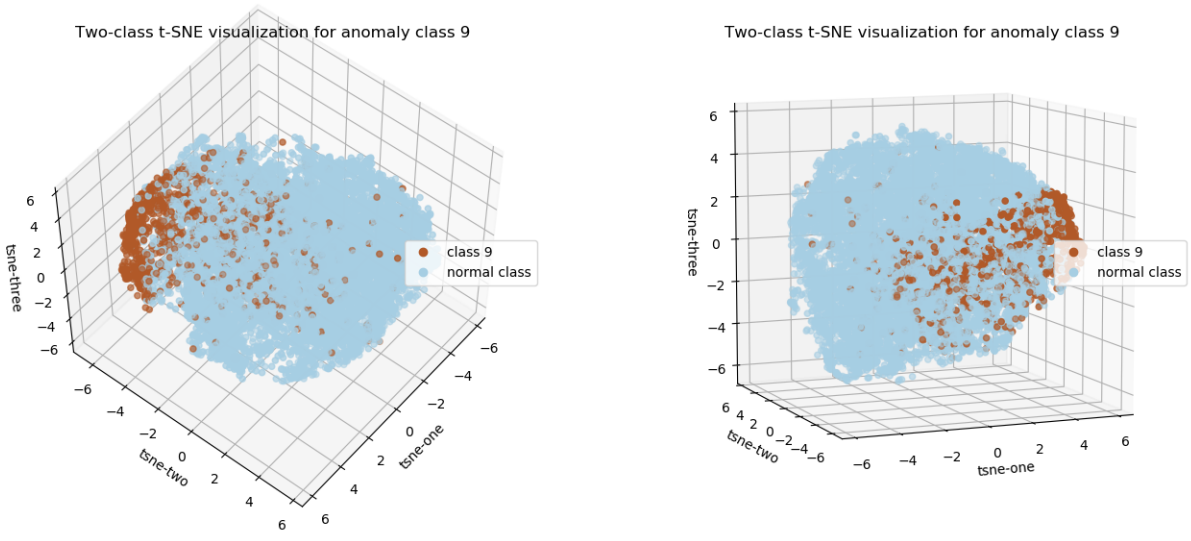
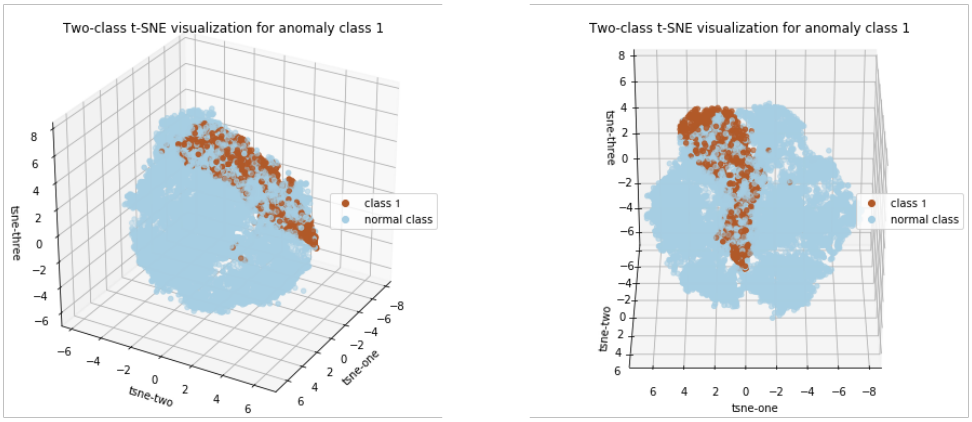


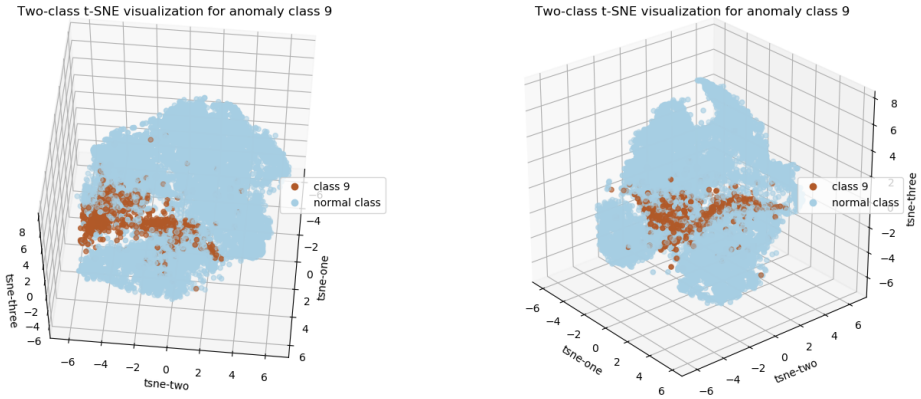
Figure 4.13: Two-class t-SNE visualization for anomaly classes 1, representation learned by a CNN with size of 8 pooling layer.

We previously hypothesised that simple shape like edge cannot be properly expressed

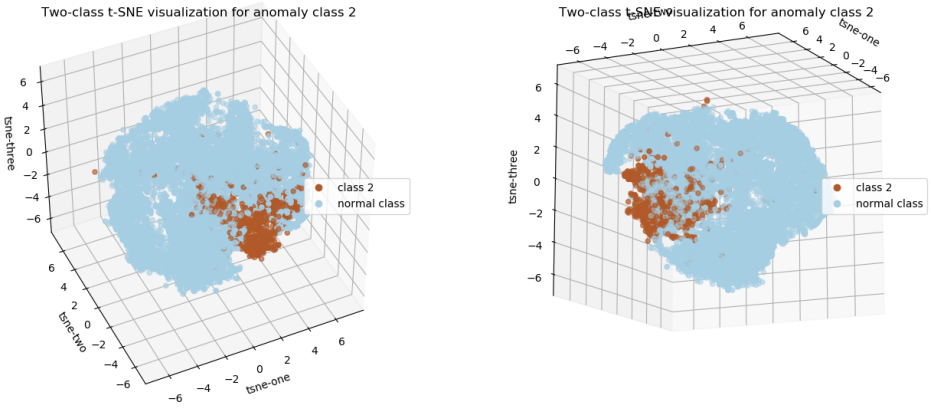
by large pooling layers. To justify this thought, we look into the data representation distribution extracted from network with small pooling layer. As shown in Figure 4.13, we discovered that class 1, the digit with the simplest shape, is presenting a good outlier's characteristic: most data are relatively distant to cluster's centre. This increase the chance of being identified as an anomaly. More distributions from PCA are located in Appendix A.



(a) Anomaly class: 1



(b) Anomaly class: 9



(c) Anomaly class: 2

Figure 4.14: Two-class t-SNE visualization for anomaly classes: 1 2 9

Chapter 5

Conclusions and Suggestions for Future Work

5.1 Report Summary

The objective of our project was to explore the relationship between size of final pooling layer for Convolutional Neural Networks and anomaly detection performance. Particularly, we examined networks' transfer learning performance on two digit dataset namely: SVHN and MNIST. Additionally, we tried to seek into the basic condition that a transfer learning task needs to meet.

In detail, we first created a pool with a total of 14 different pooling layer size CNNs. We trained them with SVHN digits then investigate these networks ability on detecting anomalous examples from MNIST dataset. Meanwhile, we also conducted a similar experiment but with dataset swapped. We repeated experiments several times until we were convinced that our results are generalized without empirical bias. The former experiment were set to explore whether or not anomaly detection performance would scale with the size of pooling layer. Based on our prior knowledge in deep learning, without any exper-

imental progress on AD, we speculated that having overly large pooling layer will cast negative impact on detecting anomalies. However, to the best extend of our experiments, our results cannot attest to our initial hypothesis, instead, performance is scalable to the size of pooling layer and is 20% better than our baseline's performance. In addition to these findings, we also noticed some anomaly digit class, e.g. 1 is inversely proportional to layer size. But such observation is statistically explicable, because digit 1 itself can be recognized as an edge which is a common feature that is shared by other classes' in their high-level representations.

Our next experiment explored the feasibility of applying transfer learning from a less varied domain to a more versatile domain. We hypothesize that this cannot work and our claim is empirically validated. Although having larger pooling layer contributes better detection result and those results are slightly better than guessing by chance or baseline algorithm, with close to 56% accuracy, we are not convinced that networks having such results is suitable for anomaly detection.

Finally, our research on visualizing data distribution really helps with unriddling puzzling finding during experiments. With most data being visualized, we are able to explain why both baseline method and CNNs act as random guess classifiers. Our analysis from Figure 4.10a and A.3 shows that our CNNs trained on MNIST cannot successfully extract peculiar features from SVHN, this is the by-product from an illy choice of dataset for transfer learning. Despite the SVHN representation learned from CNNs by MNIST indeed gives some separation boundary between classes, our anomaly score function cannot utilize such representation in practice. We therefore hypothesized that with a more advanced scoring function or through data augmentations, transfer from less varied domain might still output desirable detection results. We also remain doubted that detection performance would continue to grow without dropping, provided with more larger pooling layer size network. In sum, our research on optimizing recognition representation on detecting anomaly data

can at least show that, raising neural networks' layer size to an impractically large value would in fact, offer researchers in anomaly detection with a better performed classifier.

5.2 Suggestions for Further Work

Future work should first consider justifying our conclusions by extending to datasets of other domains such as the x-ray imagery or medical imaging dataset. Due to limited computing resources, we are unable to pursue our study with larger pooling layer, we hereby suggest prospected researchers can push the size of pooling layer into its limit to further explore whether our findings holds. For researchers with strong computational resources, it might be worth the efforts in conducting relevant experiment on CNNs and dataset from [31] which is what we originally planned to do.

Hitherto, state-of-art advancements in deep learning, it would be interesting to implement our research objective to more advanced neural-network's framework such as the Residual Neural Network [33] and Generative Adversarial Network [70].

Bibliography

[1] Aderemi O. Adewumi and Andronicus A. Akinyelu. A survey of machine-learning and nature-inspired based credit card fraud detection techniques. *International Journal of System Assurance Engineering and Management*, 8(S2):937–953, dec 2016. doi: 10.1007/s13198-016-0551-y.

[2] Jerone T. A. Andrews, Nicolas Jaccard, Thomas W. Rogers, and Lewis D. Griffin. Representation-learning for anomaly detection in complex x-ray cargo imagery. In Amit Ashok, Edward D. Franco, Michael E. Gehm, and Mark A. Neifeld, editors, *Anomaly Detection and Imaging with X-Rays (ADIX) II*. SPIE, may 2017. doi: 10.1117/12.2261101.

[3] Jerone TA Andrews, Edward J Morton, and COM Lewis D Griffin. Transfer representation-learning for anomaly detection. 2016.

[4] Jerone TA Andrews, Edward J Morton, and Lewis D Griffin. Detecting anomalous data using auto-encoders. *International Journal of Machine Learning and Computing*, 6(1):21, 2016.

[5] John E. Ball, Derek T. Anderson, and Chee Seng Chan. Comprehensive survey of deep learning in remote sensing: theories, tools, and challenges for the community. *Journal of Applied Remote Sensing*, 11(04):1, sep 2017. doi: 10.1117/1.jrs.11.042609.

[6] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35 (8):1798–1828, aug 2013. doi: 10.1109/tpami.2013.50.

[7] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[8] Mateusz Buda, Atsuto Maki, and Maciej A. Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 106: 249–259, oct 2018. doi: 10.1016/j.neunet.2018.07.011.

[9] Raghavendra Chalapathy and Sanjay Chawla. Deep learning for anomaly detection: A survey.

[10] Raghavendra Chalapathy, Aditya Krishna Menon, and Sanjay Chawla. Anomaly detection using one-class neural networks.

[11] Raghavendra Chalapathy, Ehsan Zare Borzeshi, and Massimo Piccardi. An investigation of recurrent neural architectures for drug name recognition. In *Proceedings of the Seventh International Workshop on Health Text Mining and Information Analysis*. Association for Computational Linguistics, 2016. doi: 10.18653/v1/w16-6101.

[12] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, July 2009. ISSN 0360-0300. doi: 10.1145/1541880.1541882. URL `http://doi.acm.org/10.1145/1541880.1541882`.

[13] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets.

[14] Sucheta Chauhan and Lovekesh Vig. Anomaly detection in ECG time signals via deep long short-term memory networks. In *2015 IEEE International Conference*

on Data Science and Advanced Analytics (DSAA). IEEE, oct 2015. doi: 10.1109/dsaa.2015.7344872.

[15] Nitesh V. Chawla, Nathalie Japkowicz, and Aleksander Kotcz. Editorial. *ACM SIGKDD Explorations Newsletter*, 6(1):1, jun 2004. doi: 10.1145/1007730.1007733.

[16] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995. doi: 10.1023/a:1022627411411.

[17] J.S. Cramer. The origins of logistic regression. *SSRN Electronic Journal*, 2003. doi: 10.2139/ssrn.360300.

[18] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4):303–314, dec 1989. doi: 10.1007/bf02551274.

[19] D. Dasgupta and F. Nino. A comparison of negative and positive selection algorithms in novel pattern detection. In *Smc 2000 conference proceedings. 2000 ieee international conference on systems, man and cybernetics. 'cybernetics evolving to systems, humans, organizations, and their complex interactions' (cat. no.0*, volume 1, pages 125–130 vol.1, Oct 2000. doi: 10.1109/ICSMC.2000.884976.

[20] Dipankar Dasgupta and Nivedita Majumdar. Anomaly detection in multidimensional data using negative selection algorithm. volume 2, pages 1039 – 1044, 02 2002. ISBN 0-7803-7282-4. doi: 10.1109/CEC.2002.1004386.

[21] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning.

[22] Sarah M. Erfani, Sutharshan Rajasegarar, Shanika Karunasekera, and Christopher Leckie. High-dimensional and large-scale anomaly detection using a linear one-class

SVM with deep learning. *Pattern Recognition*, 58:121–134, oct 2016. doi: 10.1016/j.patcog.2016.03.028.

[23] Eleazar Eskin, Andrew Arnold, Michael Prerau, Leonid Portnoy, and Sal Stolfo. A geometric framework for unsupervised anomaly detection. In *Applications of data mining in computer security*, pages 77–101. Springer, 2002.

[24] Zhou Fei-yan, Jin Lin-peng, and Dong Jun. Review of convolutional neural network. *Chinese Journal of Computers*, 40(6):1229–1251, 2017. (In Chinese).

[25] Gary William Flake and Barak A Pearlmutter. Differentiating functions of the jacobian with respect to the weights. In *Advances in Neural Information Processing Systems*, pages 435–441, 2000.

[26] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff. A sense of self for unix processes. In *Proceedings 1996 IEEE Symposium on Security and Privacy*, pages 120–128, May 1996. doi: 10.1109/SECPRI.1996.502675.

[27] K. Fukumizu and S. Amari. Local minima and plateaus in hierarchical structures of multilayer perceptrons. *Neural Networks*, 13(3):317–327, apr 2000. doi: 10.1016/s0893-6080(00)00009-5.

[28] A. Gibson and J. Patterson. *Deep Learning: A Practitioner's Approach*. O'Reilly, 2017. ISBN 9781491914250. URL `https://books.google.co.uk/books?id=BdPrrQEACAAJ`.

[29] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323, 2011.

[30] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[31] Lewis D. Griffin, Matthew Caldwell, Jerone T. A. Andrews, and Helene Bohler. "unexpected item in the bagging area": Anomaly detection in x-ray security images. *IEEE Transactions on Information Forensics and Security*, 14(6):1539–1553, jun 2019. doi: 10.1109/tifs.2018.2881700.

[32] Simon Hawkins, Hongxing He, Graham Williams, and Rohan Baxter. Outlier detection using replicator neural networks. In Yahiko Kambayashi, Werner Winiwarter, and Masatoshi Arikawa, editors, *Data Warehousing and Knowledge Discovery*, pages 170–180, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg. ISBN 978-3-540-46145-6.

[33] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition.

[34] G. E. Hinton. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, jul 2006. doi: 10.1126/science.1127647.

[35] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, jul 2006. doi: 10.1162/neco.2006.18.7.1527.

[36] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

[37] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, jan 1989. doi: 10.1016/0893-6080(89)90020-8.

[38] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications.

[39] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[40] Rishabh Kumar Jain, Rajeswari Ponnuru, Ajit Kumar P., and Ravi Keron N. Cifar-10 classification using intel® optimization for tensorflow. Intel® AI Developer Program, December 2017. URL `https://software.intel.com/en-us/articles/cifar-10-classification-using-intel-optimization-for-tensorflow`.

[41] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization.

[42] B. Kiran, Dilip Thomas, and Ranjith Parakkal. An overview of deep learning based methods for unsupervised and semi-supervised anomaly detection in videos. *Journal of Imaging*, 4(2):36, feb 2018. doi: 10.3390/jimaging4020036.

[43] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, may 1983. doi: 10.1126/science.220.4598.671.

[44] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[45] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, may 2017. doi: 10.1145/3065386.

[46] Donghwoon Kwon, Hyunjoo Kim, Jinoh Kim, Sang C. Suh, Ikkyun Kim, and

Kuinam J. Kim. A survey of deep learning-based network anomaly detection. *Cluster Computing*, sep 2017. doi: 10.1007/s10586-017-1117-8.

[47] Longin Jan Latecki, Aleksandar Lazarevic, and Dragoljub Pokrajac. Outlier detection with kernel density functions. In *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pages 61–75. Springer, 2007.

[48] R. Laxhammar, G. Falkman, and E. Sviestins. Anomaly detection in sea traffic - a comparison of the gaussian mixture model and the kernel density estimator. In *2009 12th International Conference on Information Fusion*, pages 756–763, July 2009.

[49] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[50] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[51] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen A.W.M. van der Laak, Bram van Ginneken, and Clara I. Sánchez. A survey on deep learning in medical image analysis. *Medical Image Analysis*, 42:60–88, dec 2017. doi: 10.1016/j.media.2017.07.005.

[52] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*. IEEE, dec 2008. doi: 10.1109/icdm.2008.17.

[53] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[54] Markus M. Breunig, Hans-Peter Kriegel, Raymond Ng, and Joerg Sander. Lof: Identifying density-based local outliers. volume 29, pages 93–104, 06 2000. doi: 10.1145/342009.335388.

[55] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3, 2013.

[56] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[57] Vijay Mahadevan, Weixin Li, Viral Bhalodia, and Nuno Vasconcelos. Anomaly detection in crowded scenes. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, jun 2010. doi: 10.1109/cvpr.2010.5539872.

[58] Stefania Matteoli, Marco Diani, and Giovanni Corsini. A tutorial overview of anomaly detection in hyperspectral images. *IEEE Aerospace and Electronic Systems Magazine*, 25(7):5–28, jul 2010. doi: 10.1109/maes.2010.5546306.

[59] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, dec 1943. doi: 10.1007/bf02478259.

[60] Goeffrey J McLachlan and GJ Mclachlan. Mahalanobis distance. *Resonance*, 4(06), 1999.

[61] Song Mei, Andrea Montanari, and Phan-Minh Nguyen. A mean field view of the landscape of two-layer neural networks. *Proceedings of the National Academy of Sciences*, 115(33):E7665–E7671, jul 2018. doi: 10.1073/pnas.1806579115.

[62] Marvin Minsky and Seymour Papert. An introduction to computational geometry. *Cambridge tiass., HIT*, 1969.

[63] Perry Moerland and Emile Fiesler. Neural network adaptations to hardware implementations. Technical report, IDIAP, 1997.

[64] Mehdi Mohammadi, Ala Al-Fuqaha, Sameh Sorour, and Mohsen Guizani. Deep learning for IoT big data and streaming analytics: A survey. *IEEE Communications Surveys & Tutorials*, 20(4):2923–2960, 2018. doi: 10.1109/comst.2018.2844341.

[65] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[66] R Okuta, Y Unno, D Nishino, S Hido, and C Loomis. Cupy: A numpy-compatible library for nvidia gpu calculations. In *of Workshop on Machine Learning Systems (LearningSys) in The Thirty-first Annual Conference on Neural Information Processing Systems (NIPS)*, 2017.

[67] Animesh Patcha and Jung-Min Park. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer Networks*, 51(12):3448–3470, aug 2007. doi: 10.1016/j.comnet.2007.02.001.

[68] Maurice Peemen, Bart Mesman, and Henk Corporaal. Efficiency optimization of trainable feature extractors for a consumer platform. In *International Conference on Advanced Concepts for Intelligent Vision Systems*, pages 293–304. Springer, 2011.

[69] Huan-Kai Peng and Radu Marculescu. Multi-scale compositionality: Identifying the compositional structures of social dynamics using deep learning. *PLOS ONE*, 10(4): e0118309, apr 2015. doi: 10.1371/journal.pone.0118309.

[70] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[71] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions.

[72] Waseem Rawat and Zenghui Wang. Deep convolutional neural networks for image classification: A comprehensive review. *Neural computation*, 29(9):2352–2449, 2017.

[73] Felix Rembold, Clement Atzberger, Igor Savin, and Oscar Rojas. Using low resolution satellite imagery for yield prediction and yield anomaly detection. *Remote Sensing*, 5(4):1704–1733, apr 2013. doi: 10.3390/rs5041704.

[74] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, sep 1951. doi: 10.1214/aoms/1177729586.

[75] Thomas W. Rogers, Nicolas Jaccard, Edward J. Morton, and Lewis D. Griffin. Automated x-ray image analysis for cargo security: Critical review and future promise. *Journal of X-ray science and technology*, 25 1:33–56, 2017.

[76] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958. doi: 10.1037/h0042519.

[77] Sebastian Ruder. An overview of gradient descent optimization algorithms.

[78] Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. Deep one-class classification. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4393–4402, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL `http://proceedings.mlr.press/v80/ruff18a.html`.

[79] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, oct 1986. doi: 10.1038/323533a0.

[80] Andrew M. Saxe, Pang Wei Koh, Zhenghao Chen, Maneesh Bhand, Bipin Suresh, and Andrew Y. Ng. On random weights and unsupervised feature learning. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pages 1089–1096, USA, 2011. Omnipress. ISBN 978-1-4503-0619-5. URL `http://dl.acm.org/citation.cfm?id=3104482.3104619`.

[81] Bernhard Schölkopf, Robert C Williamson, Alex J Smola, John Shawe-Taylor, and John C Platt. Support vector method for novelty detection. In *Advances in neural information processing systems*, pages 582–588, 2000.

[82] Bernhard Schölkopf, John C. Platt, John Shawe-Taylor, Alex J. Smola, and Robert C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1471, jul 2001. doi: 10.1162/089976601750264965.

[83] Nathan Shone, Tran Nguyen Ngoc, Vu Dinh Phai, and Qi Shi. A deep learning approach to network intrusion detection. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2(1):41–50, feb 2018. doi: 10.1109/tetci.2017.2772792.

[84] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition.

[85] Daniel Soudry and Yair Carmon. No bad local minima: Data independent training error guarantees for multilayer neural networks.

[86] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net.

[87] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15:1929–1958, 2014.

[88] Ingo Steinwart, Don Hush, and Clint Scovel. A classification framework for anomaly detection. *J. Mach. Learn. Res.*, 6:211–232, December 2005. ISSN 1532-4435. URL `http://dl.acm.org/citation.cfm?id=1046920.1058109`.

[89] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks.

[90] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*, 2015. URL `http://arxiv.org/abs/1409.4842`.

[91] Yuliya Tarabalka, Trym Vegard Haavardsholm, Ingebjørg Kåsen, and Torbjørn Skauli. Real-time anomaly detection in hyperspectral images using multivariate normal mixture models and GPU processing. *Journal of Real-Time Image Processing*, 4(3):287–300, dec 2008. doi: 10.1007/s11554-008-0105-x.

[92] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[93] Yung Liang Tong. *The multivariate normal distribution*. Springer Science & Business Media, 2012.

[94] Aaron Tuor, Samuel Kaplan, Brian Hutchinson, Nicole Nichols, and Sean Robinson. Deep learning for unsupervised insider threat detection in structured cybersecurity data streams. 2017.

[95] Kalyan Veeramachaneni, Ignacio Arnaldo, Vamsi Korrapati, Constantinos Bassias, and Ke Li. AIˆ2: Training a big data machine to defend. In *2016 IEEE 2nd International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS)*. IEEE, apr 2016. doi: 10.1109/bigdatasecurity-hpsc-ids.2016.79.

[96] H. Vogel and D. Haller. Luggage and shipped goods. *European Journal of Radiology*, 63(2):242–253, aug 2007. doi: 10.1016/j.ejrad.2007.03.040.

[97] Shui-Hua Wang, Khan Muhammad, Jin Hong, Arun Kumar Sangaiah, and Yu-Dong Zhang. Alcoholism identification via convolutional neural network based on parametric relu, dropout, and batch normalization. *Neural Computing and Applications*, Dec 2018. ISSN 1433-3058. doi: 10.1007/s00521-018-3924-0. URL `https://doi.org/10.1007/s00521-018-3924-0`.

[98] Eric W. Weisstein. Wolfram imageidentify net v1. From MathWorld—A Wolfram Web Resource. URL `\url{https://resources.wolframcloud.com/NeuralNetRepository/resources/Wolfram-ImageIdentify-Net-V1}`. Last visited on 1/4/2019.

[99] K. Wells and D.A. Bradley. A review of x-ray explosives detection techniques for checked baggage. *Applied Radiation and Isotopes*, 70(8):1729–1746, aug 2012. doi: 10.1016/j.apradiso.2012.01.011.

[100] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 2(1-3):37–52, aug 1987. doi: 10.1016/0169-7439(87)80084-9.

[101] Ruobing Wu, Baoyuan Wang, Wenping Wang, and Yizhou Yu. Harvesting discriminative meta objects with deep CNN features for scene classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE, dec 2015. doi: 10.1109/iccv.2015.152.

[102] Kwang Eui Yoo and Youn Chul Choi. Analytic hierarchy process approach for identifying relative importance of factors to improve passenger security checks at airports. *Journal of Air Transport Management*, 12(3):135–142, may 2006. doi: 10.1016/j.jairtraman.2005.11.006.

[103] Yufeng Zheng and Adel Elmaghraby. A vehicle threat detection system using correlation analysis and synthesized x-ray images. In J. Thomas Broach and Jason C. Isaacs, editors, *Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XVIII*. SPIE, jun 2013. doi: 10.1117/12.2016646.

Appendix A

Supplementing Results

Figure A.1 gives the intermediate result when using the right value of ϵ for our baseline methods.

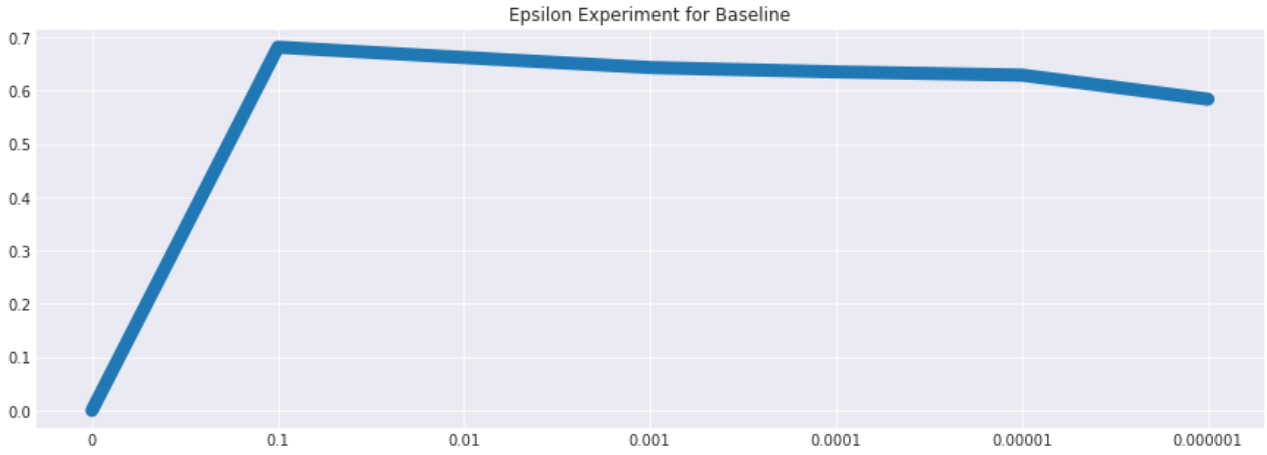


Figure A.1: Impact of tunning epsilons for baseline

Figure A.2 showed our baseline result in 10 runs. We can see our baseline AD performance is closed to 50*chance*.

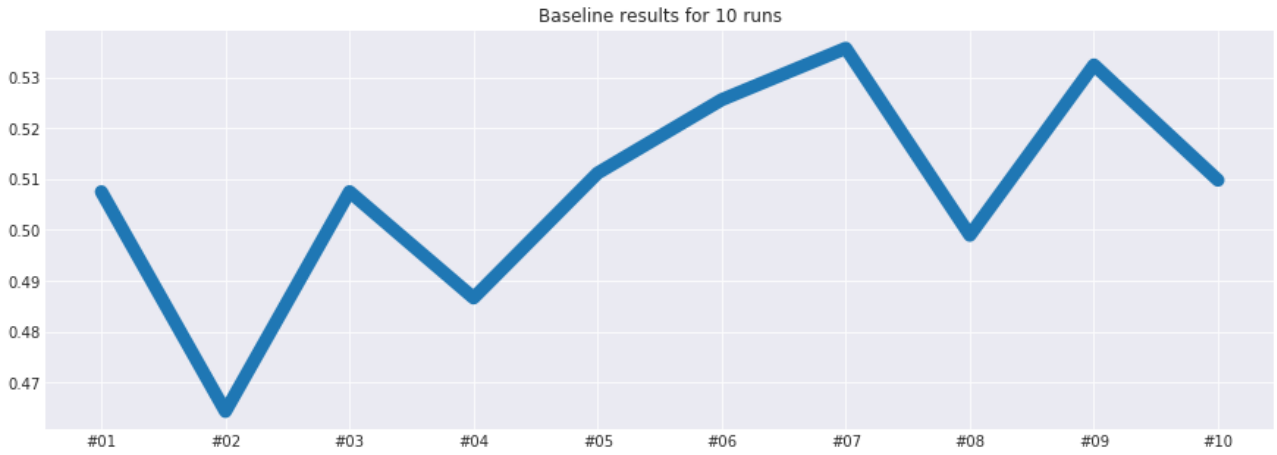


Figure A.2: Results of baseline for SVHN AD over 10 runs

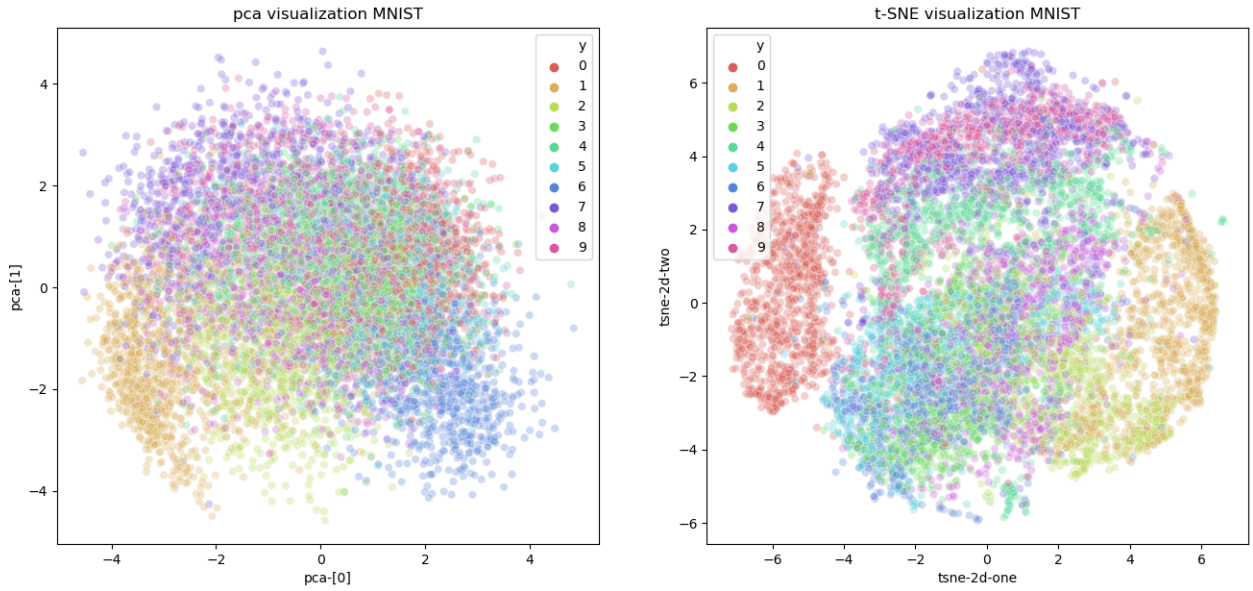


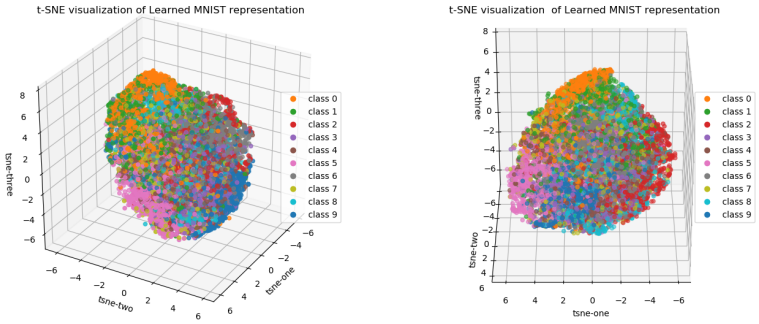
Figure A.3: Representation distribution for SVHN from CNNs trained on MNIST.

Throughout our training process, we plot how learned representation from MNIST are distributed with varied pooling layers. In Figure A.4

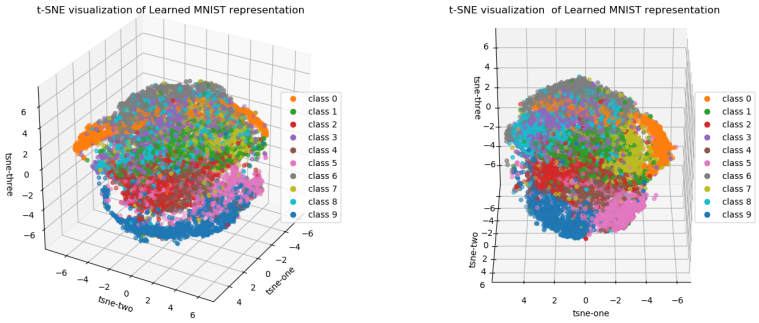
It is clear that cluster of each digits are becoming further away as we increase the pooling layer size this means, final pooling layer size for CNNs, in general, are positively

correlations to anomaly detection performance when transfer learning is applicable.

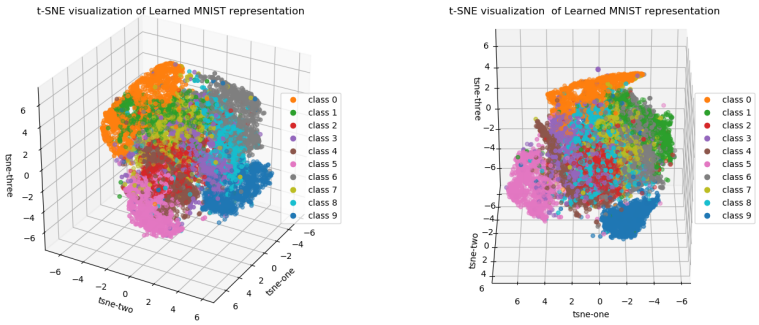
When trying to visualize digit representation's distribution with PCA given in Figure A.5, we notice the data for class 1 and 9 in space spanned by principal components are embedded into the cluster, but graph from t-SNE is possible to disentangle these data, therefore we decide to include graph only from t-SNE to main body.



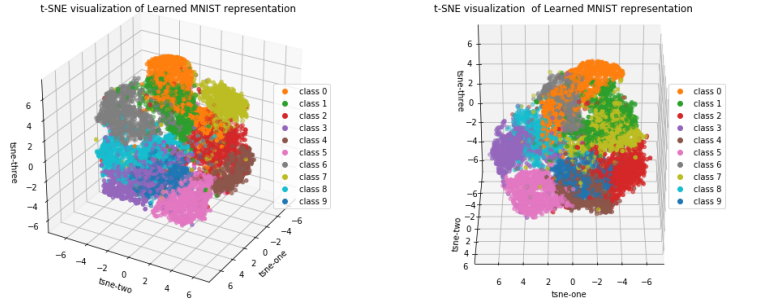
(a) Final pooling layer size: 8



(b) Final pooling layer size: 128



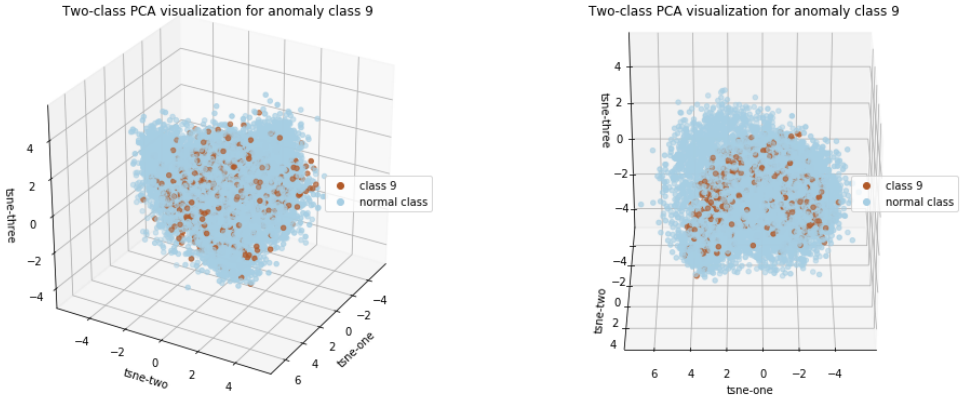
(c) Final pooling layer size: 256



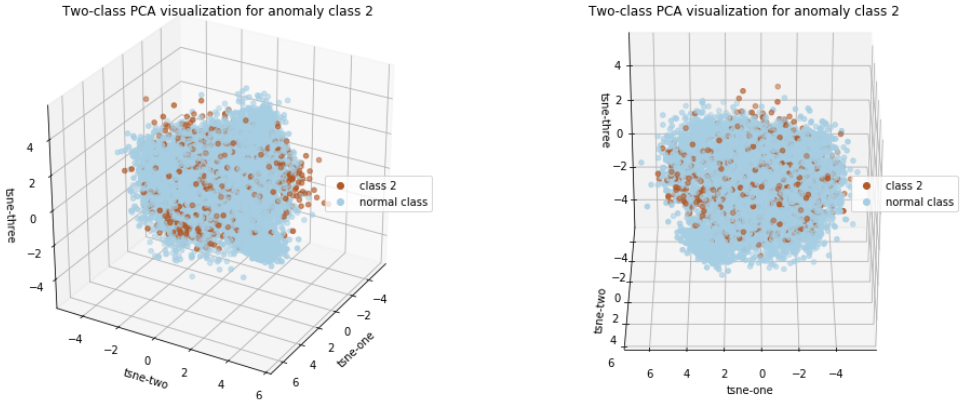
(d) Final pooling layer size: 512

Figure A.4: Visualization of how CNNs learn representations. Dataset: MNIST

(a) Anomaly digit: 1



(b) anomaly digit: 9



(c) anomaly digit: 2

Figure A.5: PCA plots for anomaly digit: 1 9 and 2.